ICASE

MULTI-LEVEL ADAPTIVE FINITE ELEMENT METHODS

I. VARIATIONAL PROBLEMS

Achi Brandt

ICASE REPORT NO. 79-8

April 2, 1979

INSTITUTE FOR COMPUTER APPLICATIONS IN SCIENCE AND ENGINEERING
NASA Langley Research Center, Hampton, Virginia

Operated by the

UNIVERSITIES SPACE USRA RESEARCH ASSOCIATION

DISPLAY 90/2/1

80N23060*# ISSUE 13 PAGE 1766 CATEGORY 64 RPT#: NASA-TM-80985

ICASE-79-8 CNT#: NAS1-14101 79/03/02 74 PAGES UNCLASSIFIED DOCUMENT

UTTL: Multi-level adaptive finite element methods. 1: Variation problems

AUTH: A/BRANDT, A.

CORP: National Aeronautics and Space Administration. Langley Research Center, Hampton, VA. AVAIL.NTIS

SAP: HC A04/MF A01

CIO: UNITED STATES

MAJS: /*DIFFERENTIAL EQUATIONS/*FINITE ELEMENT METHOD/*INTEGRAL EQUATIONS/* PROBLEM SOLVING

MINS: / BOUNDARY VALUE PROBLEMS/ DISCRETE FUNCTIONS/ EXTRAPOLATION/ RELAXATION METHOD (MATHEMATICS)/ TIME DEPENDENCE

ABA: M.G.

ABS: A general numerical strategy for solving partial differential equations and other functional problems by cycling between coarser and finer levels of discretization is described. Optimal discretization schemes are provided together with very fast general solvers. It is described in terms of finite element discretizations of general nonlinear minimization problems. The basic processes (relaxation sweeps, fine-grid-to-coarse-grid transfers of residuals, coarse-to-fine interpolations of corrections) are directly and naturally determined by the objective functional and the sequence of approximation spaces. The natural processes, however, are not

ENTER: MORE

4B - EM3287.PRT

MULTI-LEVEL ADAPTIVE FINITE-ELEMENT METHODS

I. VARIATIONAL PROBLEMS

A. Brandt

Department of Applied Mathematics
The Weizmann Institute of Science
Rehovot, Israel

ABSTRACT

The Multi-Level Adaptive Technique (MLAT) is a general numerical strategy of solving partial-differential equations and other functional problems by cycling between coarser and finer levels of discretization. It provides nearly optimal discretization schemes together with very fast general solvers. It is described here in terms of finite element discretizations of general non-linear minimization problems. The basic processes (relaxation sweeps, fine-grid-to-coarse-grid transfers of residuals, coarse-to-fine interpolations of corrections) are directly and naturally determined by the objective functional and the sequence of approximation spaces. The natural processes, however, are not always optimal. Concrete examples are given and some new techniques are reviewed, including the local truncation extrapolation and a multi-level procedure for inexpensively solving chains of many boundary-value problems, such as those arising in the solution of time-dependent problems.

Part of the work reported here was performed under NASA Contract No. NAS1-14101 while the author was in residence at ICASE, NASA Langley Research Center, Hampton, Virginia 23665.

TABLE OF CONTENTS

PART 1.

VARIATIONAL PROBLEMS

1. INTRODUCTION

The Multi-Level Adaptive Technique (MLAT) is a general numerical strategy for solving continuous problems such as differential and integral equations and functional optimization problems.

In most numerical procedures for solving such problems, the analyst first discretizes the problem, choosing approximating algebraic equations on a finite dimensional approximation space, and then devises a numerical process to (nearly) solve this huge system of discrete equations. Usually, no real interplay is allowed between discretization and solution processes. This results in enormous waste: the discretization process, being unable to predict the proper resolution and the proper order of approximation at each location, uses an approximation space which is too fine. The algebraic system thus becomes unnecessarily large in size, while accuracy usually remains rather low, since local smoothness of the solution is not being properly exploited. On the other hand, the solution process fails to take advantage of the fact that the algebraic system to be solved does not stand by itself, but is actually an approximation to continuous equations, and therefore can itself be approximated by other (much simpler) algebraic systems.

The basic idea of adaptive processes is that an efficient discretization of a problem depends on the solution itself: A smooth solution can be approximated in a coarse approximation space (a space with relatively few degrees of freedom, such as those produced by coarse triangulations). The coarse approximation can in fact be very good, provided its order is sufficiently high (e.g., provided finite elements are used which contain all

polynomials up to a sufficiently high degree). A highly-oscillating solution, by contrast, can be approximated only in an approximation space which is fine enough to resolve the oscillations. In general the solution may be smooth in one subdomain, oscillating in another, and may have all kinds of singularities around some special points or manifolds. In each region then, the efficient discretization will be different in nature. Thus, finding the efficient discretization becomes an integral part of the problem. The problem is therefore solved iteratively, and at certain stages the discretization is adapted to the evolving solution.

The multi-level techniques go one step further by recognizing that it is not necessary, at every stage, to adapt the discretization to the solution; it is enough, and much more efficient, to adapt the discretization to the error in the solution. A smooth error can very efficiently be liquidated by a coarse-space approximation. The fine (and computationally expensive) approximation spaces are needed only for approximating the highly-oscillating part of the solution; they should be used only to smooth out oscillating errors. Smoothing the error is much less expensive than liquidating it, because it can be done locally. For example, the process of relaxation is a local process which very efficiently smooths the error, but which, due to its local character, is very slow in liquidating smooth errors.

Thus, the multi-level adaptive technique is to use not a single but a sequence of approximation spaces (levels), with geometrically decreasing mesh-sizes. New levels may be introduced and changed in the process, and they constantly interact with each other. The solution procedure involves

relaxation sweeps over each level, coarse-level-to-fine-level interpolations of corrections, and fine-to-coarse transfers of residuals. This procedure has two important basic benefits: On the one hand it acts as a very general fast solver of the discrete system of equations. On the other hand it provides, in a natural way, a flexible and adaptive discretization. The total number n of discrete variables can thus be kept low, and the solution of the n algebraic equations is obtained in a low number of operations. In fact, the number of operations is only $O(n)$, since a couple of relaxation sweeps at the finest level are enough to make the error so smooth that the rest of the work can be done on coarser levels.

MLAT have so far been developed mainly in finite-difference formulations. A comprehensive description of this development, together with historical notes, can be found in [B3]. For a simplified survey of ideas and software, see [B4]. Some further developments are reported in [B5], [B6], [B7], and [D1]. Throughout that development close agreement has been obtained between theoretical predictions and numerical experiments. Both show that all boundary-value finite-difference problems considered can be solved by the multi-grid algorithm in 5 to 10 work units, where a work unit is defined as the computational work involved in processing all the difference equations of the finest level one time (i.e., the work in one simple relaxation sweep over the finest level). Problems ranged from the simplest model problem, through singular pertubation problems, to fairly complicated nonlinear Navier-Stokes equations and transonic flows, with a variety of domains and boundary conditions. The more recent developments include methods for time-dependent problems, very efficient multi-grid embedding processes (e.g., for bifurcation problems) and methods for local-truncation extrapolations (cf. respectively Sections 3.9, 3.8, and 3.10 below).

The special capability of the finite-difference multi-level structure to create non-uniform, flexible discretization patterns based on uniform grids is discussed in Section 3 of [B5]. This capability is obtained by observing that the various uniform grids (levels) need not all extend over the entire domain. Finer levels may be confined to increasingly smaller subdomains, so as to provide higher resolution only where desired. Moreover, one may attach to each of these localized finer grids its own local system of coordinates, to fit curved boundaries or to approximate directions of interior interfaces and thin layers. All these patches of local grids interact with each other through the multi-grid process, which provides fast solution to their difference equations.

That structure is very flexible. Since it is based on uniform grids, it is feasible to employ high-order difference approximations, and to change the order whenever desired, with no extra investment in computational work. The effective (the finest) mesh-size can be locally adjusted in any desired pattern by changing (extending or contracting) the various uniform grids, expending negligible amounts of bookkeeping work and storage. These flexibilities of a finite difference technique pose a challenge to the finite element method, which traditionally had the edge in terms of flexibility, especially in treating curved boundaries.

It has been theoretically shown in [B3] and [B5] that, using this flexible structure in an adaptive solution process governed by certain criteria, one can get exponential rates of convergence. That is, the error (in solving the <u>differential</u> problem) decreases exponentially as a function of the computational work. Such rates are obtainable even for problems with singular pertubations, algebraic singularities, corners, etc.

Parallel to the above investigations, an interest has been growing in applying multi-level techniques in finite element formulations. Such applications are mentioned in [B2] and briefly described in various sections of [B3]. The purpose of the present article is to give a full description of the multi-level adaptive techniques in terms of finite elements for general types of problems, and to discuss their various algorithmic and theoretical aspects.

At the same time other workers, starting with Nicolaides [N1], [N2] and then Hackbusch [H1] - [H3], Bank and Dupont [BD] and Mansfield [M1] have established the mathematical foundations of certain multi-grid finite elements algorithms. For a growing class of problems they have rigorously proven the basic multi-grid assertion, namely, that the discrete algebraic problem with n degrees of freedom (obtained as a finite element approximation to a continuous problem) can be solved in only Cn computer operations; or, at least, that $Cn \log \frac{1}{\epsilon}$ operations are enough to reduce the L^2 norm of the error by any desired factor ϵ. The constant C is independent of n, but its numerical value is usually not specified. In fact, if numerical values of C were calculated from the rigorous proofs, they would turn out to be exceedingly large, much larger for example than practical values of n, so that for practical purposes the Cn estimates would look worse than some $\bar{C}n^2$ estimates. (The only exceptions are the rigorous estimates in Appendix C of [B3] and a similar result in [F1]. But they apply to the model problem only.)

Such rigorous investigations are of a very different character than the present work. The price of rigor is that the results are far from realistic. The proofs give meaningful estimates only for extremely large

n, and, even then, the work estimates are orders of magnitude too large. The estimates are therefore too crude to yield any practical information. e.g., they cannot resolve the difference between more efficient and less efficient multi-grid processes. (This difference is crucial in practice; it may itself cover several orders of magnitude.) For this reason, and since the quantity we try to estimate here is actually nothing but the computer time (which of course we know anyway, at least aposteriori), a different type of theoretical studies are preferred by the present author.

Discarding rigor, our studies are based on the observation that the important multi-level processes are of a local nature, since low-frequency corrections are obtained by coarse-level processes, which cost very little. One can therefore analyze the crucial aspects of multi-grid processes by employing a local mode analysis: Far boundaries and low-frequency changes in the coefficients of the equations can be ignored, so that the effect of multi-level processes on individual Fourier components can easily be calculated. General computer routines have been developed to perform this analysis automatically for any given problem, yielding precise quantitative predictions of the multi-level efficiency. Experiments with various types of equations (see [D1] and [P1]) show the work predictions to be precise within a few percent. This tool (combined with related observations, the most important of which is the "coupled nonsmoothness" mentioned in Section 3.2 below) is therefore useful in selecting efficient algorithms (see, e.g., Section 6 and Appendix A in [B3]), in understanding the numerical results, and in debugging multi-level programs (see Section 3.7.9 below). It played an important role in developing to full efficiency

many multi-grid finite difference codes.

The multi-grid experience with finite-difference formulations is of course very relevant to finite element ones, since usually finite element methods give a type of finite-difference equations. It is, in fact, the only experience we have. Namely, all numerical experiments so far, even those based on finite element derivations (see [N3] and [P1]), were based on uniform grids (e.g., uniform triangulations), so that finite-difference structure of the discrete equations was very explicit. But there are some special features in finite element formulations which do not show clearly in general finite difference equations. (The converse is also true.)

The special features of the finite element method show themselves most naturally in variational problems, of course. We start therefore our study of multi-level finite element methods with the general minimization problem, where the entire description is given in terms of the "total energy" functional E, the minimization of which is our objective. It is not assumed that E is quadratic, nor that it has any other special form, so that the described method is applicable to general nonlinear problems. The basic multi-level processes (relaxation sweeps, fine-to-coarse transfers of residuals and coarse-to-fine interpolations of corrections) turn out to be completely natural; they are directly determined by the objective functional E and the approximation spaces. In particular we reproduce, for the general nonlinear case, the observation (made first by Nicolaides, see [N1]) that the natural residual transfer is the adjoint of the correction interpolation. Convergence is guaranteed since E decreases monotonically by each step of relaxation, as well as by the coarse-level corrections.

A closer examination reveals that the natural processes are not always the best ones. For example, the natural finite-element interpolation from

a coarse approximation space to a fine one (e.g., the identity interpolation in case the fine space contains the coarse one) is sometimes too crude (see Section 3.1). In some other cases, interpolations cruder (i.e., of lower order) than the natural one could be used without loss of efficiency (see Section 3.3). The τ-extrapolation technique, which can improve very much the nodal values of the approximation (Section 3.10), is better understood from a finite-difference point of view. In fact, the extrapolation does not considerably improve the quality of the approximation when measured in terms of the usual finite element norm. All these findings are related of course to the known fact that the point-wise errors of the finite elements are quite often much larger than the average error. While this fact has little effect in usual finite element algorithms, it is important in the multi-level processes, which should take advantage of the relations between coarse and fine discretizations. Nevertheless, the natural processes are not bad, and can safely be used, even though they may sometimes require an order of magnitude more computing time.

A central issue in finite element methods is how important it is to use uniform subdivisions. The nonuniform elements are important on the boundaries, but it seems that in the interior there is usually no need for nonuniformity, while uniformity offers substantial gains in computing time and storage. See for example the uniform interior structure in Figure 2 in Section 4.2 below. Uniformity becomes even more advantageous when multi-level solution methods are used (one reason being that, since the solution of the discrete equations is much faster, it becomes more important to speed up other parts of the algorithm, in particular the assembly of the stiffness equations, which is very laborious when nonuniform elements are used). Moreover, the multi-level technique, as mentioned above, has its own mechanism,

and a very efficient one, for creating nonuniform discretizations, based on a collection of uniform grids. We will discuss this issue in a later part of this paper.

In the later part of the paper some generalizations will be given. Multi-level solutions to general Galerkin (weak form) finite element discretizations will be described. This will include new types of relaxation (e.g., distributive relaxation, as mentioned in [B5], [B6], and [B7]) and a further study of the relation between interpolations and residual weightings. More general problems will be included, such as constrained minimization problems, degenerate problems, indefinite and non-elliptic problems, and eigenvalue problems. Local mode analysis and adaptation techniques will be treated in greater detail.

2. MINIMIZATION PROBLEM AND APPROXIMATION SPACES

2.1 The Continuous Problem.

We consider a d-dimensional variational problem of minimizing a functional $E(u)$ over an admissible space S of functions $u(x)$, where $x = (x_1,\ldots,x_d)$ and u may be a vector of functions $(u_{(1)},\ldots,u_{(q)})$. E need not be quadratic, but we assume that E and S are such that a unique minimum, denoted* U, exists, at least locally. Thus, our problem is to find U such that

$$E(U) = \min_{u\in S} E(u) . \qquad (2.1)$$

For some standard examples the reader is referred to Chapter 4 below.

* Generally, we will use capital letters to denote solutions; the corresponding lower-case letters will denote approximations.

2.2 Approximation Spaces.

An approximation space S^ℓ is a finite-dimensional space which contains approximations to U. It is convenient to assume that S^ℓ is a subspace of S (although this is not absolutely necessary). The dimension of S^ℓ is n_ℓ, and its basis functions are $\varphi_j^\ell(x)$, $(j=1,\ldots,n_\ell)$:

$$S^\ell = \{u^\ell \mid u^\ell(x) = \sum_{j=1}^{n} u_j^\ell \varphi_j^\ell(x)\}, \tag{2.2}$$

where u_j^ℓ are the nodal values of the trial function u^ℓ. Usually, with each approximation space S^ℓ, we associate a real parameter h_ℓ representing its typical mesh-size, i.e., the support of each basis function φ_j^ℓ is assumed to have linear dimensions comparable to h_ℓ.

We may like sometimes to regard S^ℓ as a tensor product of q_ℓ spaces $S_1^\ell,\ldots,S_{q_\ell}^\ell$, i.e.,

$$u^\ell = (u_{[1]}^\ell, u_{[2]}^\ell, \ldots, u_{[q_\ell]}^\ell).$$

Thus, u_j^ℓ denotes the j-th nodal-value, $u_{(\alpha)}^\ell$ is the α-th component-function of u^ℓ in S, while $u_{[\beta]}^\ell$ is the β-th component-function of u^ℓ in S^ℓ. The functions $u_{[\alpha]}^\ell$ need not correspond to $u_{(\alpha)}^\ell$. In fact, sometimes $q_\ell > q$, for example, when u^ℓ approximates derivatives of u as well as point-values of u itself:

$$u_j^\ell = \frac{\partial^\nu}{\partial x_1^{\nu_1}, \ldots, \partial x_d^{\nu_d}} u_{(\alpha)}^\ell(x). \tag{2.3}$$

In such cases, we denote α, $x=(x_1,\ldots,x_d)$, $\nu=\nu_1+\ldots+\nu_d$ and $\underline{\nu}=(\nu_1,\ldots,\nu_d)$ by α_j^ℓ, $x_j^\ell=(x_{j1}^\ell,\ldots,x_{jd}^\ell)$, $\nu_j^\ell=\nu_{j1}^\ell+\ldots+\nu_{jd}^\ell$ and $\underline{\nu}_j^\ell=(\nu_{j1}^\ell,\ldots,\nu_{jd}^\ell)$, respectively. A function $u_{[\beta]}^\ell$ is formed by $\Sigma u_j^\ell \varphi_j^\ell$, summing over

all j with the same α_j^ℓ and the same $\underline{\nu}_j^\ell$. More generally, $u_{[\beta]}^\ell$ can be formed by summing $u_j^\ell \varphi_j^\ell$ over all j for which u_j^ℓ represents the same linear combination of expressions like (2.3). The distinction between different functions $u_{[\beta]}^\ell$ is not essential to multi-level processes, and is not really used below. It can be useful, however, in devising the interpolation routines and in theoretical discussions.

When u_j^ℓ satisfy (2.3), the approximation elements are called nodal finite elements ([SF], p. 101).

Multi-level solution processes typically employ a sequence of increasingly finer approximation spaces $S^1, S^2, \ldots, S^M$, with corresponding mesh-sizes $h_1 > h_2 > \ldots > h_M$, and usually $h_\ell = 2h_{\ell+1}$. S^M may be the "target space", i.e., the space in which the final approximation is sought, with coarser spaces $S^1, S^2, \ldots, S^{M-1}$ serving only as auxiliary spaces. In adaptive processes, however, there is of course no target-space and the relation between spaces will become more involved.

The simplest case is that of the nested approximation spaces

$$S^1 \subset S^2 \subset \ldots \subset S^M \subset S. \tag{2.4}$$

This case may arise if each element of $S^{\ell-1}$ is a union of S^ℓ-elements. In most applications, however, such a requirement, especially near boundaries, would pose a severe limitation. Thus, in our general description we do not assume (2.4), but the reader may like to keep this case in mind as a simple example.

2.3 Interpolation Between Approximation Spaces and Relative Smoothness.

In multi-level processing we will need interpolation operators I_k^ℓ which will be linear transformations from S^k to S^ℓ such that $I_k^\ell u^k$ is "close" to u^k. In the nested case $S^k \subset S^\ell$ it is natural to take I_k^ℓ as the identity operator I (although this is not always the best choice). Generally, the usual polynomial interpolation procedures, of suitable order, can be employed to obtain nodal values of $I_k^\ell u^k$ which approximate u^k as well as possible (i.e., to the order of the best approximation to u^k from S^ℓ). The order of interpolation may vary by circumstances, and will be further discussed below (e.g., Section 3.1). At this point we only introduce, as a general characterization of I_k^ℓ, the matrix $I^{k\ell}$ defined by

$$I_k^\ell \varphi_i^k = \sum_j I_{ij}^{k\ell} \varphi_j^\ell \ . \tag{2.5}$$

Let S^k be a space coarser than S^ℓ. The smoothness of a function $u^\ell \in S^\ell$ relative to S^k is measured by the quantity

$$1 - \inf_{u^k \in S^k} \frac{\| I_k^\ell u^k - u^\ell \|}{\| u^\ell \|} \ . \tag{2.5a}$$

Thus, the notion of relative smoothness depends on the interpolation operator I_k^ℓ. For example, in nodal finite elements (2.3), if polynomial interpolation is applied independently for each function $u_{[\beta]}^k$, relative smoothness is measured in terms of the smoothness of the individual functions $u_{[\beta]}^\ell$. A function u^ℓ may be smooth in this sense, without being smooth in the usual sense, e.g., without having smooth component-functions $u_{(\alpha)}^\ell$. Normally, however, our interpolation operator I_k^ℓ will be the natural one, and the smoothness (2.5a) will be equivalent to the smoothness of $u_{(\alpha)}^\ell$.

2.4 Derivatives of Functionals.

Let $E(u^\ell)$ be any functional defined on S^ℓ. We define its derivative with respect to the j-th nodal value by

$$E^\ell_j(u^\ell) = \lim_{\delta\to 0} \frac{1}{\delta} \{E(u^\ell + \delta\ \varphi^\ell_j) - E(u^\ell)\} . \tag{2.6}$$

Similarly, we may use the notation $E^\ell_{ij} = (E^\ell_i)^\ell_j$, and also define derivatives with respect to nodal values of another space S^k:

$$E^k_i(u^\ell) = \lim_{\delta\to 0} \frac{1}{\delta} \{E(u^\ell + \delta\, I^\ell_k \varphi^k_i) - E(u^\ell)\} . \tag{2.7}$$

If $u^\ell = u^k \in S^k \subset S^\ell$, then $E^k_i(u^\ell) = E^k_i(u^k)$. From (2.5) and (2.7), it follows that if E^ℓ_j are continuous, then

$$E^k_i(u^\ell) = \sum_j I^{k\ell}_{ij}\, E^\ell_j(u^\ell) . \tag{2.8}$$

Orders in h. Note that E^ℓ_j is not fully equivalent to the first variation (Frechet derivative) of E in S. It is proportional to the "volume" $|\varphi^\ell_j|$ of the basis function φ^ℓ_j. For example, for nodal finite elements (2.3), if u^ℓ is a smooth function, then $E^\ell_j(u^\ell) = O(h^d)$.

2.5 Ritz Discretizations.

As our discrete approximation to (2.1) on S^ℓ, we have the problem of finding $U^\ell \in S^\ell$ such that

$$E(U^\ell) = \min_{u^\ell \in S^\ell} E(u^\ell) . \tag{2.9}$$

This implies the "stiffness equations"

$$E_j^\ell(U^\ell) = 0 , \quad (j = 1,2,\ldots,n_\ell) . \tag{2.10}$$

This is a system of n_ℓ equations for the n_ℓ unknowns (the nodal values of U^ℓ).

The (discrete, as well as the original) problem is called linear if the system (2.10) is a linear system, that is, if E is quadratic, or equivalently, if the stiffness coefficients $E_{ij}^\ell(u^\ell)$ are constants independent of u^ℓ. The system can then be rewritten as

$$\sum_i E_{ji}^\ell U_i^\ell = E_j^\ell(0) . \tag{2.11}$$

In multi-level processing, use is made of the residuals of an approximation u^ℓ: the j-th residual is the value of $-E_j^\ell(u^\ell)$, which expresses by how much u^ℓ fails to satisfy the discrete equation (2.10). The discrete solution U^ℓ has zero residuals.

It is important to notice the different scales of the discrete equations (2.10) at different levels ℓ. In finite difference methods we write the discrete equations as analogs of the differential equations, and as a result, the equations have the same scale on all levels. By this we mean that the residuals of any smooth function are all O(1) in terms of the mesh-size h. This

is no longer so in (2.10) where the residuals of smooth functions are proportional to $|\varphi_j^\ell|$, which is often $0(h_\ell^d)$.

In the case of nodal finite elements (2.3), equations (2.10) will usually take the form of difference equations. We then denote by $m_{(\alpha)}$ and $m_{[\beta]}$ the highest order of differences applied in (2.10) to the functions $u_{(\alpha)}^\ell$ and $u_{[\beta]}^\ell$, respectively. Hence, the ratio between the residuals of smooth and nonsmooth functions $u_{(\alpha)}^\ell$ is $0(h_\ell^{m(\alpha)})$, and for $u_{[\beta]}^\ell$ the ratio is $0(h_\ell^{m[\beta]})$.

Assembling stiffness matrices for multi-level processing can be made by the usual procedures. The need for assembly for several approximation spaces adds only a small amount to the programming effort and costs in computer time and space only a fraction more than the finest-level assembly. (cf. Sec. 3.7.8.)

3. MULTI-LEVEL SOLUTION PROCESSES

We consider first the usual situation where some "target" approximation space S^M is given in which the solution of the stiffness equations (2.10, ℓ=M) is sought. In this section we describe processes which provide fast solution to this algebraic system, whether linear or nonlinear, by the iterative usage of some given coarser approximation spaces $S^1, S^2, \ldots, S^{M-1}$. The same processes will later be shown to be the main building blocks in more developed adaptive procedures, when no target approximation spaces are set in advance.

3.1 Coarse-Space Approximation.

Let S^k be an approximation space coarser than S^ℓ ; namely, h_k is considerably larger than h_ℓ (typically, $h_k = 2\, h_\ell$) so that $n_k << n_\ell$. Solving the S^k stiffness equations, by any method, is therefore much less expensive than solving the S^ℓ equations. The simplest and most familiar way of using S^k in the iterative solution of the S^ℓ equations is to interpolate an (approximate) solution from S^k to S^ℓ , to serve as a first approximation u^ℓ :

$$u^\ell \leftarrow I_k^\ell\, u^k \quad , \quad (u^k \text{ approximates } U^k) \ . \tag{3.1}$$

How good this first approximation is depends on the smoothness of the solution U^ℓ . In some cases U^ℓ is so smooth that, if the interpolation I_k^ℓ is of order high enough to exploit the smoothness, then the first approximation (3.1) will turn out to be good enough and will require no further improvement. In such cases, however, the approximation space S^ℓ is not really needed: S^k already yields the solution to the required accuracy. Thus, if the S^ℓ approximation is at all needed, the first approximation (3.1) will require a considerable improvement.

Can we compute a correction to u^ℓ again by the inexpensive use of the coarse space S^k ? Namely, can we somehow approximate the error $V^\ell = U^\ell - u^\ell$ by some $V^k \in S^k$? Normally*, the answer is no. If u^k in (3.1) is a good enough approximation to U^k , then V^ℓ will be a rapidly oscillating function that cannot meaningfully be approximated in the coarse space S^k . Therefore, before we can reuse coarse spaces, the error V^ℓ

*An exception is the case where S^k does not fully use the smoothness of the solution U . In that case, if I_k^ℓ in (3.1) is of sufficiently high order, then V^ℓ will be smooth enough to be approximated by some $V^k \in S^k$. This situation is related, however to the inappropriate approximation order being used, and will therefore not arise in a fully adaptive procedure.

should be smoothed out. Smoothing and coarse-space corrections are described in the next sections. Here we add some remarks on the first coarse-space approximation (3.1).

The question arises, of what order and what types should the interpolation (3.1) be? If $S^k \subset S^\ell$, a natural choice is the identity $u^\ell = u^k$. Usually this choice is not the best: u^k has an error with large rapid oscillations (cf., e.g., [SF] p. 168). Such an error, as we will see below, is the most expensive to liquidate by multi-level processes. Hence, a very substantial gain can be made by producing the nodal values of u^ℓ through a higher-order polynomial interpolation, which will give much smaller rapid error oscillations. For best results, the rapid error oscillations generated by interpolation should not be larger than the rapid (i.e., wavelength $O(h_k)$) oscillations expected anyway in the solution U itself. Hence, for best results, the interpolation order should be high enough to exploit all the expected smoothness of the solution U. The most significant part of this gain will already be obtained if the interpolation order is such that the order of magnitude of the residuals which are produced is not larger than that of the local truncation errors (see Section 3.5 below). In other words, the interpolation should be exact for every polynomial which minimizes a functional E both in S and in S^ℓ (see the example in Section 4.2). In particular, interpolation of nodal values of $u^\ell_{(\alpha)}$ should be of order $m_{(\alpha)}+p$ (i.e., should use polynomials of degree $m_{(\alpha)}+p-1$), where p is the order of approximation. For analyses of interpolation orders see Section A.2 in [B3].

Another possibility is to use two coarse-space solutions, $u^k \in S^k$ and $u^j \in S^j$ say, with mesh-sizes $h_j > k_k$, and to define u^ℓ by h-extrapolation from u^j and u^k. This is a reasonable procedure only when S^k and S^j have uniform grids. Even then, the accuracy obtained is at best equivalent to that obtained by $\bar{S}^j$, a space with mesh-size h_j but with a higher order of approximation. In principle, it is less expensive to solve the $\bar{S}^j$ problem than that of S^k (since $h_j > h_k$). Thus, in a fully adaptive process, a situation where h-extrapolation can profitably be used will not arise.

3.2 Error Smoothing by Relaxation.

A basic solution process, used in multi-level algorithms mainly as an error smoothing process, is the relaxation sweep. The simplest example (and in a sense also the best) is the following:

Gauss-Seidel relaxation sweep: This is a process in which all the nodal values of some given approximate solution u^ℓ are scanned one by one in some prescribed order. Each nodal value u_j^ℓ, in its turn, is replaced by a new value $\bar{u}_j^\ell$, which is the value for which the energy (2.10) will be as small as possible, other nodal values being held fixed. In other words, $\bar{u}_j^\ell$ is chosen so that the corresponding stiffness equation (2.10) is satisfied.

Actually, if this equation is non-linear in u_j^ℓ, it is better not to satisfy it completely, but to make only one Newton step toward its solution. Namely,

$$\bar{u}_j^\ell = u_j^\ell - E_j^\ell(u^{\ell,j}) \,/\, E_{jj}^\ell(u^{\ell,j}) \quad , \tag{3.2}$$

where $u^{\ell,j}$ is the approximate solution just prior to replacing u_j^ℓ by $\bar{u}_j^\ell$. (That is, if in our prescribed order i is scanned before j, then $u_i^{\ell,j} = \bar{u}_i^\ell$, otherwise $u_i^{\ell,j} = u_i^\ell$.)

Having completed a relaxation sweep, the system (2.10) is not yet solved, of course, because its equations are coupled to each other. A well known and extensively used method for solving sparse algebraic systems like (2.10) is by a long sequence of relaxation sweeps. This is a convergent process, since $E(u^\ell)$ is monotonically decreasing. But the rate of convergence is very slow. Typically, if n_ℓ is the dimension of S^ℓ, then the number of sweeps required for convergences increases as n_ℓ^α (see [Y1]).

A closer examination, however, reveals that the convergence is not slow as long as the error $V^\ell = U^\ell - u^\ell$ has rapid oscillations (oscillations with wave-

length comparable to h_ℓ). Such error oscillations are typically reduced at least by a factor .5 per sweep. Thus, the convergence slows down only when v^ℓ becomes smooth.

In other words, relaxation sweeps, inefficient as they are in solving problems, are very efficient in smoothing out the error. This property, which will be extensively used below, is very general for Gauss-Seidel relaxation of any non-degenerate discretization of a minimization problem. Degeneracy occurs when the stiffness system of equations is decomposable, at least locally, into several decoupled (or weakly coupled) systems. In such cases efficient smoothing can still be obtained by more sophisticated Gauss-Seidel relaxation schemes, which take the degeneracy into account, e.g., line relaxation in suitable directions.

Remark: For a system ($q_\ell > 1$), the relaxation may be slowly converging in some, but not all, the components. For example, in the "mixed method", when both $u^\ell_{(\alpha)}$ and some of its derivatives are taken as independent unknowns, relaxing over the equations related to derivatives (i.e., relaxing $E^\ell_j(u^\ell)=0$ over the j for which $\nu = \nu^\ell_j$ 0 in (2.3)) will converge very fast. This of course does not help, since the lowest-order equations converge slowly; but it stresses the fact that these lowest-order equations should be the primal concern in the steps below (coarse-grid corrections). These equations are also the ones which produce the finite difference analog to the differential equations. If there are several zero-order unknowns (u^ℓ_j such that $\nu^\ell_j=0$) per mesh cube, the finite difference analog may be produced only by combinations of the corresponding equations. Those special combinations will then be slow to converge, so it is no help that some other combinations may converge rapidly by suitable relaxation sweeps.

Another remark: There are other relaxation schemes which are quite natural to the minimization problem. The most obvious one is the steepest descent method, in which all the unknowns are changed simultaneously, each one in proportion to the decrease in E per its unit change. In terms of difference equations this method is known as Jacobi (under) relaxation. The Gauss-Seidel relaxation is usually preferable, since it requires less storage and provides better smoothing rates. Jacobi relaxation, however, is more suitable for parallel computations.

A final remark about relaxation: the efficient smoothing process does not continue indefinitely. Except for some ideal cases (e.g., equations with constant coefficients in the infinite domain, uniformly triangularized and consistently relaxed), a certain level of rapid error oscillations is always coupled to the smooth errors. Starting from a completely smooth error function, rapid error oscillations are generated by the relaxation sweeps because of boundary interaction and variations in the stiffness coefficients. In particular, relaxing with highly oscillatory coefficients will produce in $U^{\ell}_{(\alpha)}$ rapid error oscillations of magnitude $O(h_{\ell}^{m(\alpha)})$ times the magnitude of the smooth error. This level of "coupled nonsmoothness" will persist as relaxation slows down. Further relaxation sweeps will be wasteful. Moreover, if the error is smoother than this level (see footnote in Section 3.1), relaxation may even magnify the rapid error oscillations instead of reducing them. Hence, in such cases it is best to avoid relaxation altogether (cf. analysis in Section A.2 of [B3]).

3.3 Measuring Dynamic Residuals.

In some multi-level algorithms we may wish to measure the current error $v^\ell = u^\ell - u^\ell$ in order to detect either convergence or slow convergence rates. v^ℓ cannot be measured directly. Instead, we can measure the residuals $-E_j^\ell(u^\ell)$. Computing all these residuals is quite expensive, however. It costs roughly as much as a relaxation sweep, so that measuring them after every sweep (as required by some algorithms) would double our computational work. Therefore, instead of computing the "static" residuals $-E_j^\ell(u^\ell)$, one usually calculates the "dynamic" residuals $-E_j^\ell(u^{\ell,j})$, which are less expensive since they are computed anyway in the course of each relaxation sweep (cf. (3.2)).

It is important that the norms in measuring the residuals on different spaces S^ℓ are comparable. That is, they must all be discrete approximations to the same continuous norm of the energy first variation. Thus, the L_2 norm of the dynamic residuals is given by

$$\| \text{res.} \|_2^\ell = \left\{ \sum_{j=1}^{n_\ell} h_\ell^d \left[|\varphi_j^\ell|^{-1} \; E_j^\ell(u^{\ell,j}) \right]^2 \right\}^{1/2} \tag{3.3}$$

(cf. Section 2.5), while their L_∞ norm is

$$\| \text{res.} \|_\infty^\ell = \max \; |\varphi_j^\ell|^{-1} |E_j^\ell(u^{\ell,j})| \; , \tag{3.4}$$

and the weighted L_1 norm (perhaps the most useful one) is

$$\| \text{res.} \|_G^\ell = \sum_{j=1}^{n_\ell} G(x_j^\ell) \, h_\ell^d \, |\varphi_j^\ell|^{-1} |E_j^\ell(u^{\ell,j})| \; . \tag{3.5}$$

One (or several) of these norms may be calculated along with each relaxation sweep.

For algorithmic decisions a dynamic-residuals norm is as good as the static one, because (i) fast relaxation convergence must exhibit a fast decrease of the dynamic norm; and (ii) when the convergence is slow the dynamic norm is equal to approximately twice the static norm (since the equations are approximately symmetric).

3.4 Coarse-Space Correction

As soon as the error $v^\ell = u^\ell - u^\ell$ in S^ℓ has been smoothed out by relaxation, a good approximation to it can be inexpensively computed in a coarser space. This idea was used for finite difference equations by Southwell [S1] and by others (cf. the historical notes in [B3]), and it has a central role in multi-level solution processes.

Let S^k be an approximation space coarser than S^ℓ. We wish to write equations for $v^k \in S^k$ so that its interpolant $I_k^\ell v^{k\ell} \in S^\ell$ will approximate v^ℓ as well as possible. Since v^ℓ is the solution of the problem

$$E(u^\ell+v^\ell) = \min_{v^\ell \in S^\ell} E(u^\ell+v^\ell) ,$$

a natural definition of v^k is by the requirement

$$E(u^\ell+I_k^\ell v^k) = \min_{v^k \in S^k} E(u^\ell+I_k^\ell v^k) , \tag{3.6}$$

which yields the equations (cf. Section 2.4)

$$E_i^{k\ell}(u^\ell+I_k^\ell v^k) = 0 , \quad (i = 1,\ldots,n_k) . \tag{3.7}$$

This is a system of n_k equations for the n_k nodal values of v^k.

For general nonlinear problems, or in the case of non-nested spaces ($S^\ell \not\supset S^k$), equations (3.7) are more complicated than necessary. They require a special assembly procedure, and the scheme may become complicated in later stages (when still coarser spaces are used in solving (3.7); see Section 3.5). To simplify the scheme we first rewrite (3.7) in the form

$$E_i^{k\ell}(u^\ell+I_k^\ell v^k) - E_i^{k\ell}(u^\ell) = -E_i^{k\ell}(u^\ell) . \tag{3.8}$$

Since v^k is smooth, the left-hand side of (3.8) can be approximated by

$$E_i^k(I_\ell^k u^\ell+v^k) - E_i^k(I_\ell^k u^\ell) . \tag{3.9}$$

Observe that we cannot similarly approximate the left-hand side of (3.7)

since $I_k^\ell v^k$ may be small compared to the rapidly oscillating part of u^ℓ, and therefore the error of the coarse approximation may be large compared with $I_k^\ell v^k$. In (3.8), by contrast, even if $I_k^\ell v^k$ is small, the left-hand side is still an approximately linear operator in $I_k^\ell v^k$. Indeed, for linear problems and nested spaces $(S^\ell \supset S^k)$, the left-hand side of (3.8) exactly coincides with (3.9).

Using (2.8) for the right-hand side of (3.8) our new equations become

$$E_i^k(I_\ell^k u^\ell + v^k) - E_i^k(I_\ell^k u^\ell) = -\sum_j I_{ij}^{k\ell} F_j^\ell(u^\ell) \quad . \tag{3.10}$$

If the problem is linear (E quadratic) these equations can be rewritten as a linear system for v^k :

$$\sum_\alpha E_{i\alpha}^k v_\alpha^k = -\sum_j I_{ij}^{k\ell} E_j^\ell(u^\ell) \quad . \tag{3.11}$$

Observe that this problem is of the same form as the usual S^k stiffness equations ((2.10) or (2.11), for $\ell = k$), except that the original right-hand side is replaced by linear combinations of residuals from the finer space S^ℓ. Thus, no special assembly is needed for these equations.

More generally, for nonlinear problems, we introduce the notation

$$\bar{u}^k = I_\ell^k u^\ell + v^k \quad , \tag{3.12}$$

in terms of which (3.10) is written as

$$E_i^k(\bar{u}^k) = E_i^k(I_\ell^k u^\ell) - \sum_j I_{ij}^{k\ell} E_j^\ell(u^\ell) \quad . \tag{3.13}$$

Again these equations are the usual S^k stiffness equations ((2.10) for $\ell = k$), only the right-hand side is new, so that no special assembly is needed.

The mode of working directly with the coarse-space correction v^k and solving (3.11) is called the Correction Scheme (CS). The mode of operating with the full approximation (basic approximation plus the correction) $\bar{u}^k$ and

solving (3.13) is called the Full-Approximation Scheme (FAS). Note that $\bar{U}^k$, as defined by (3.12), depends on u^ℓ. If $I_\ell^k u^\ell = u^\ell$, then $\bar{U}^k$ coincides with U^k (the solution of (2.10) for $k=\ell$). At convergence, however, $u^\ell = U^\ell$ and $V^\ell = 0$, hence

$$\bar{U}^k = I_\ell^k U^\ell \qquad \text{(at convergence)} . \tag{3.14}$$

Thus $\bar{U}^k$ is a coarse-space function which interpolates a finer-space solution, and therefore, its nodal values have a finer-space accuracy. This situation can be exploited extensively (cf. Section 2 in [B5]), so that the FAS is not only more general than CS, but it also offers other advantages, and is therefore preferable even in many linear problems.

Let v^k be an approximate solution to (3.11), or $\bar{u}^k$ an approximate solution to (3.13), obtained by a method to be specified below (Section 3.6). In the first (CS) case, the computed correction should simply be interpolated to S^ℓ and used to correct u^ℓ, namely

$$u^\ell_{NEW} \leftarrow u^\ell_{OLD} + I_k^\ell v^k . \tag{3.15}$$

In the second (FAS) case, it is important to realize that $\bar{u}^k$ itself does not approximate a smooth function that can profitably be interpolated to S^ℓ. It is $v^k = \bar{u}^k - I_\ell^k u^\ell$, approximating the smooth function V^ℓ, which we should interpolate. That is, in FAS,

$$u^\ell_{NEW} \leftarrow u^\ell_{OLD} + I_k^\ell (\bar{u}^k - I_\ell^k u^\ell_{OLD}) . \tag{3.16}$$

Note that $I_k^\ell I_\ell^k$ is not the identity operator, hence (3.16) is not equivalent to simply $u^\ell_{NEW} \leftarrow I_k^\ell u^k$. This latter interpolation would destroy the higher-frequency content of u^ℓ.

The correction (3.15) or (3.16) is called the S^k correction to u^ℓ. The smooth part of the error V^ℓ practically disappears by such a correction. High-

frequency error components are introduced by the I_k^ℓ interpolation, but they are easily liquidated by subsequent relaxation sweeps. Before turning to algorithmic details, some remarks should be added concerning the interpolation symbols above.

Three interpolation processes were used. The correction interpolation (I_k^ℓ in (3.15) or (3.16)), the residual weighting (right-hand side of (3.11) and a similar term in (3.13)), and the FAS solution averaging (I_ℓ^k in (3.13) and (3.16)). We use the term "averaging" for interpolations from a finer space to a coarser one, where the coarse space values are obtained as weighted averages of fine-space values.

The correction interpolation in terms of the above notation is given by

$$(I_k^\ell v^k)_j = \sum_l v_l^k I_{lj}^{k\ell} . \qquad (3.17)$$

It is thus clear from (3.13) that the residual-weighting is determined by (and actually is the adjoint of) the correction interpolation. For linear problems this was observed in [N1]. We will see later that in more general formulations the choice of residual weighting is independent of the correction interpolation, but in the present case (minimization problems), the relation between the two is natural and need not be violated. Sometimes, if relaxation smooths the residual function as efficiently as it smooths v^ℓ, simpler and less work-consuming residuals weighting can be used without degrading the coarse-space correction (see Section 4.2 below), but the possible gain is quite marginal and should not be attempted without specific knowledge.

The correction interpolation itself can be made in several ways. In the nested case ($S^\ell \supset S^k$), it can simply be the identity. Unlike the first coarse-

to-fine interpolation (Section 3.1), higher-order interpolation is not normally* needed here, since at this stage the smooth error components are no longer as dominant. The natural interpolation (transfer of point-wise values of S^k) is of high enough order to get the full benefit of the coarse-space correction. In fact, if high-order elements are used, lower-order interpolations could sometimes be used, with no loss in efficiency. The interpolation order need only be high enough so as not to generate high-frequency residuals larger than the low-frequency residuals of the interpolated correction. (See more on interpolation orders in Section A.2 of [33] and in the chapter on Galerkin formulations.)

The form of the FAS solution-averaging I_ℓ^k is immaterial as long as u^ℓ is smooth. Care, however, should be taken when u^ℓ has wild oscillations on the scale of the grid, i.e., when $\max_{|x-y|=h} |u^\ell(x) - u^\ell(y)|$ is comparable to $\| u^\ell \|$. To see this we can view the (nonlinear) equations (2.10) as quasi-linear equations whose coefficients depend on the solution. In the case of wild oscillations in u^ℓ, the values of $I_\ell^k u^\ell$, and hence also the values of the coefficients in the coarse-grid equations (3.13), depend very much on the form of I_ℓ^k. In such a case the fine-grid difference operator may have wildly-oscillating coefficients, and the coarse-grid operator needs to represent a proper "homogenization" (cf. Babuska (1975) and Spagnolo (1975)). For the purposes of multi-grid processes, enough homogenization is obtained if I_ℓ^k is a "full" averaging operator, i.e., any local operator such that

$$\int I_\ell^k w^\ell dx \approx \int w^\ell dx \quad \text{for any } w^\ell \in S^\ell . \tag{3.18}$$

---

*An exception is the case mentioned in footnote in Section 3.1, in which further gain is obtained by using higher-order interpolation for the next coarse-space correction.

3.5 Relative Local Truncation Errors.

Let I^ℓ denote an interpolation (projection) from the solution space S to the approximation space S^ℓ. The "local truncation errors" of the approximation space S^ℓ are the values

$$\bar{\tau}_j^\ell = E_j^\ell(I^\ell U) \tag{3.19}$$

which are (up to a sign) the residuals of the true solution U. They are so named because they serve as a measure for how well the continuous problem is locally approximated by the discrete system (2.10). Approximate knowledge of the truncation errors is important for various algorithmic criteria, such as, discretization adaptation criteria and natural stopping criteria. When the residuals $-E_j^\ell(u^\ell)$ are small in magnitude compared with the corresponding truncation errors, then U^ℓ is approximated by u^ℓ better than by the exact solution U. Hence, at that instance, u^ℓ need no further improvement, and the iterative solution of the stiffness equations may be terminated.

Moreover, if $\bar{\tau}_j^\ell$ were known, we could improve the discrete equations. In fact, replacing the discrete equations (2.10) by

$$E_j^\ell(U^\ell) = \bar{\tau}_j^\ell \ , \tag{3.20}$$

we would get the solution $U^\ell = I^\ell U$, that is, a solution which coincides, up to the interpolation I^ℓ, with the true differential solution U. For example, if I^ℓ is a point-wise projection, the nodal values of U^ℓ would coincide with those of U.

Of course, since U is unknown, the truncation errors are not known either. Consider, however, the situation described above where we had two approximation spaces, coarser space S^k and finer one S^ℓ. Since U^ℓ is usually much closer to U than to U^k, we can approximate the S^k local

truncation errors $\bar{\tau}_i^k$ by the values

$$\bar{\tau}_i^{k\ell} = E_i^k(I_\ell^k U^\ell) \ . \tag{3.21}$$

These values are called the <u>local truncation errors of S^k relative to S^ℓ</u> .

These values too are not really known until U^ℓ is fully calculated. We may, however, replace U^ℓ in (3.21) by its evolving approximation u^ℓ . More precisely, if $U^\ell - u^\ell$ is a smooth function, then

$$\begin{aligned} E_i^k(I_\ell^k U^\ell) - E_i^k(I_\ell^k u^\ell) &\approx E_i^k(U^\ell) - E_i^k(u^\ell) \\ &= -E_i^k(u^\ell) \end{aligned}$$

and hence

$$\bar{\tau}_i^{k\ell} \approx \tau_i^{k\ell} \ ,$$

where

$$\begin{aligned} \tau_i^{k\ell} &= E_i^k(I_\ell^k u^\ell) - E_i^k(u^\ell) \\ &= E_i^k(I_\ell^k u^\ell) - \sum_j I_{ij}^{k\ell} E_j^\ell(u^\ell) \ . \end{aligned} \tag{3.22}$$

At convergence, when $u^\ell = U^\ell$, we indeed get $\bar{\tau}_i^{k\ell} = \tau_i^{k\ell}$.

The approximate relative-truncation-error $\tau_i^{k\ell}$ is exactly the right-hand side of the coarse-space (S^k) correction equations (3.13), which may therefore be rewritten as

$$E_i^k(\bar{U}^k) = \tau_i^{k\ell} \ . \tag{3.22a}$$

We may thus view the role of the fine space (S^ℓ) as serving to improve, or to correct, the coarse-space equation ((2.10) for $\ell = k$) by adding to it an approximation of the local truncation error. While adding the true truncation error $\bar{\tau}_i^k$ to the coarse-space equations would make their solution U^k coincide with the true solution (i.e., $U^k = I^k U$), adding the relative truncation error $\bar{\tau}_i^{k\ell}$ makes U^k coincide with the fine-space solution

(i.e., $U^k = I_\ell^k U^\ell$) .

$\tau_l^{k\ell}$ is a satisfactory approximation to $\bar{\tau}_l^{k\ell}$ as soon as

$$\| \tau^{k\ell} - \bar{\tau}^{k\ell} \| << \| \bar{\tau}^{k\ell} \| . \tag{3.23}$$

It has been observed in the numerical experiments that (3.23) is already easily satisfied at the stage of the first transition from the S^ℓ relaxation sweeps back to S^k . Heuristically, this is explained as follows: u^ℓ at that stage was obtained by the interpolation (3.1) followed by relaxation. The interpolation leaves a residuals function $E^\ell(u^\ell)$ which is comparable to $\bar{\tau}^{k\ell}$ both in its low-frequency components (this is trivial) and in its high-frequency components (this is obtained if a suitable order of interpolation is used, cf. Section 3.1). The relaxation sweeps considerably reduce the high-frequency residual components. The bulk of the remaining low-frequency (smooth) error in $\bar{\tau}^{k\ell} - \tau^{k\ell}$ is then reduced by adding to $\tau_l^{k\ell}$ the last term in (3.2 Thus, a relation like (3.23) is satisfied by all components, regardless of the norm that is used. In fact, at the said stage, $\| \tau^{k\ell} - \bar{\tau}^{k\ell} \|$ will usually already be comparable to $\| \tau^\ell \|$; further reduction of $\| \tau^{k\ell} - \bar{\tau}^{k\ell} \|$ is not meaningful.

A sequence of refinements. Assume we have a sequence of increasingly finer approximations $u^1 \in S^1$, $u^2 \in S^2$,..., $u^\ell \in S^\ell$. The corrected coarse-grid equations (3.22a) would take the form*

$$E_l^k(U^k) = \tau_l^k \tag{3.24a}$$

where $\tau_l^k = \tau_l^{k\ell}$ if $k = \ell-1$. For $k < \ell-1$, however, the correction τ_l^k should correct the S^k equation not by the original S^{k+1} equation, but by

*We drop the bar from U^k . Hereinafter, wherever a finer-space approximation u^{k+1} exists, U^k will denote the solution of the correction equation (3.24). The original equation (2.10) is the special case $\tau_l^\ell = 0$.

its own corrected form. Hence, generally (cf. (3.22))

$$\tau_i^k = E_i^k(I_{k+1}^k u^{k+1}) - \sum_j I_{ij}^{k,k+1}\left\{E_j^{k+1}(u^{k+1}) - \tau_j^{k+1}\right\}, \quad (1 \le k \le \ell-1) \qquad (3.24b)$$

where

$$\tau_i^\ell = 0 \ . \qquad (3.24c)$$

At convergence, the solutions of (3.24) on all levels coincide with each other, namely,

$$U^k = I_{k+1}^k U^k = I_{k+1}^k I_{k+2}^{k+1} \cdots I_\ell^{\ell-1} U^\ell \qquad \text{(at convergence)} \ .$$

Thus, each τ_i^k represents the local truncation error on S^k relative to the finest level S^ℓ .

Orders in h . By usual Taylor expansions it is easy to find that the local truncation error satisfies

$$\bar{\tau}_j^\ell = b(x_j^\ell, U) h_\ell^{\gamma_j^\ell} + O(h_\ell^{\bar{\gamma}_j^\ell}) \ , \qquad (3.25a)$$

where the coefficient b depends on the local properties of the exact solution U but not on the mesh-size h_k , and $\bar{\gamma}_j^\ell > \gamma_j^\ell$ (usually $\bar{\gamma}_j^\ell = \gamma_j^\ell + 1$ or $\bar{\gamma}_j^\ell = \gamma_j^\ell + 2$). γ_j^ℓ is the local order of the truncation error. For example, in the simple nodal case ((2.3), with $\nu_j^\ell = 0$ and $q_\ell = 1$) one finds that $\gamma_j^\ell = d+p$, where d , the dimension, enters as the scale of the stiffness equation (cf. Sec. 2.5) and p is the order of approximation (which usually means that the elements contain all polynomials of degree less than p , but not all those of degree p).

When there are more discrete functions ($q_\ell > 1$) then γ_j^ℓ becomes more complicated, and may actually depend on j . If S^ℓ and S^k have the same structure, and if i in S^k corresponds to j in S^ℓ (i.e., $x_i^k = x_j^\ell$ and $\nu_i^k = \nu_j^\ell$), then they share the same constant in (3.25a), and hence

$$\bar{\tau}_j^{\ell} = \left(\frac{h_\ell}{h_k}\right)^{\gamma} \bar{\tau}_i^{k} \ (1 + o(1)) \ , \ (\gamma = \gamma_i^k = \gamma_j^\ell) \quad . \tag{3.25b}$$

Similarly, we can get a relation of the form

$$\bar{\tau}_i^{k\ell} = C_j^{k\ell} \ \bar{\tau}_i^{k} \ (1 + o(1)) \quad , \tag{3.25c}$$

where $C_j^{k\ell}$ is a known quantity (independent of U). For example, in the simple nodal case we have

$$\begin{aligned} \bar{\tau}_i^{k\ell} &\approx \bar{\tau}_i^{k} - \left(\frac{h_k}{h_\ell}\right)^{d} \bar{\tau}_j^{\ell} \\ &\approx \left\{1 - \left(\frac{h_\ell}{h_k}\right)^{p}\right\} \bar{\tau}_i^{k} \quad . \end{aligned} \tag{3.25d}$$

3.6 Multi-Level Algorithm.

By combining relaxation sweeps that smooth out the error in u^{ℓ} with coarse-space corrections that liquidate smooth errors, all error components can be efficiently reduced. The question remains of how to solve the coarse-grid problem (3.13). This is done by a similar process of relaxation sweeps (over u^k) combined with still-coarser-space corrections.

Thus, in multi-level solution processes, a sequence of approximation spaces $S^1, S^2, \ldots, S^M$ is used, starting with a very coarse space S^1, and ending with the target fine space S^M. The typical mesh-size of the space S^k is $h_k \approx 2^{-k} h_0$. Often, the triangulation for S^k is based on vertical and horizontal grid lines, and the grid lines of S^k are every other line of S^{k+1}.

Multi-grid algorithms work themselves up from the coarsest level S^1 to the finest S^M. We will denote by ℓ the current finest level, that is, the largest k for which an approximate solution u^k has already been computed. For each ℓ, a first approximation u^{ℓ} is obtained by interpolating from $u^{\ell-1}$, and then it is improved by relaxation sweeps and coarse-space corrections, using equations (3.24) throughout. One type of multi-level algorithm flows as follows (see Figure 1).

A. Solving on the coarsest grid. Set $\ell = 1$. Compute an approximate solution u^1 to the stiffness equations (3.24) on the coarsest grid ($k = \ell = 1$), either by relaxation or by some direct method. (The term direct method here means a non-iterative solution of linear systems. If the stiffness equations are nonlinear, the direct method will include a few Newton iterations, where the linear system at each iteration is solved

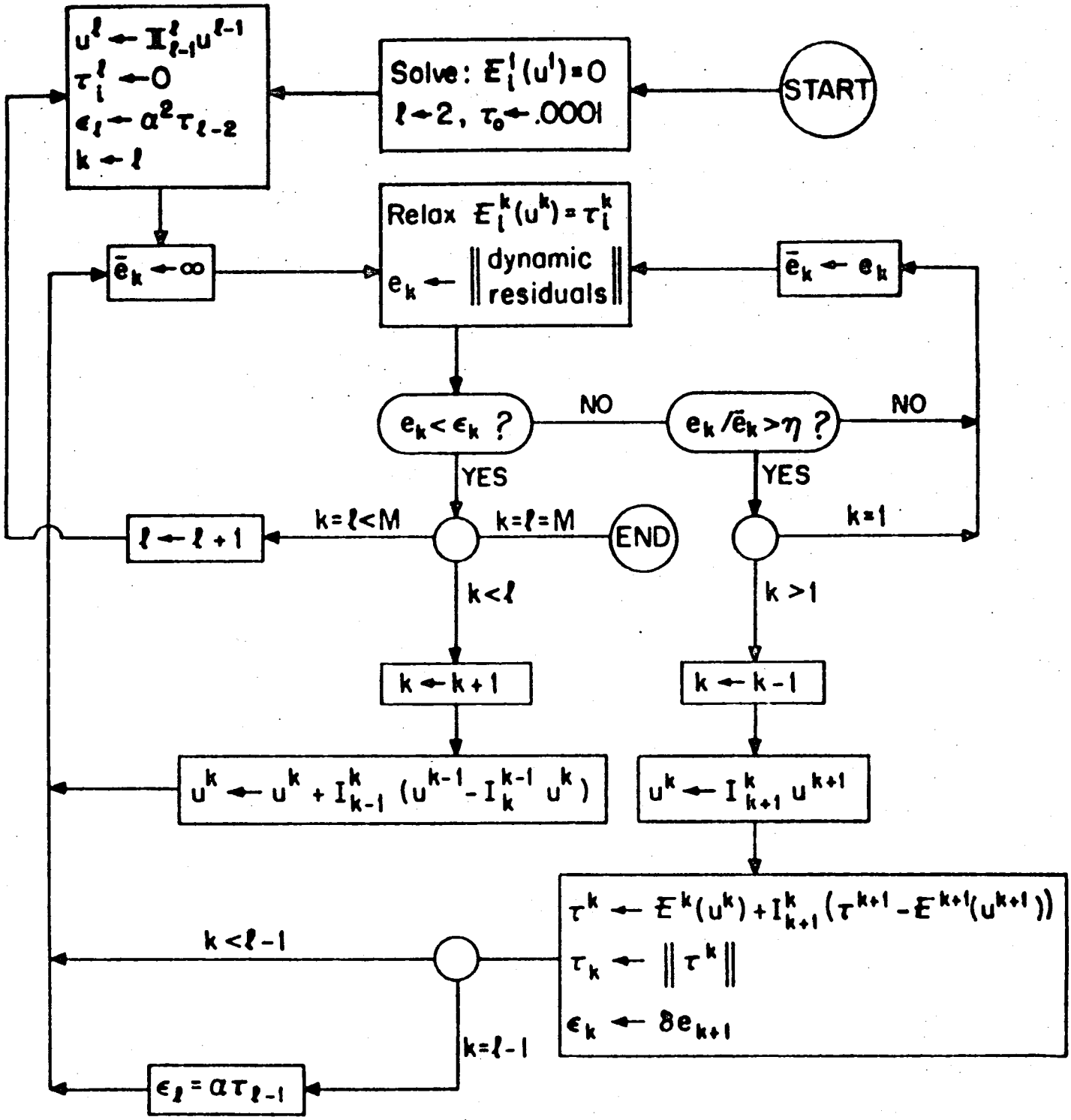


Figure 1. FAS Multi-Level Algorithm

The notation is generally explained in the text. I_{l-1}^{l} *denotes higher-order (see Section 3.1) interpolation, while* I_{k-1}^{k} *is the "natural" interpolation.* τ_k *denotes* $\| \tau_i^k \|$. *In this flowchart the coarsest-level* (k=1) *correction equations are solved by relaxation.*

directly.) Whatever the method, u^1 should be easy to obtain, since S^1 is very coarse.

B. Setting a new finest level. If $\ell = M$ the algorithm is terminated. If not, increase ℓ by 1. Introduce, as the first approximation for the new finest level, the function

$$u^\ell = I^\ell_{\ell-1} u^{\ell-1}$$

where the interpolation is the higher-order one (see Section 3.1). Having assembled the stiffness equations (2.10) for this new level, set $k = \ell$. Generally, k will denote the current operation level, (thus, for example, when the algorithm later switches to coarser-space corrections, we will have $k < \ell$) and $u^k \in S^k$ will denote the current approximation on that level. Also set ε_ℓ sufficiently small. ε_k will generally be used as the tolerance for solving the k-level equations (3.24). For ε_ℓ, a realistic value is introduced in Step G below, so the current "sufficiently small" value is only temporary.

C. Starting a new operation level. When we start working on any level k, we put $\bar{e}_k = +\infty$, for reasons to become clear in Step E below.

D. Relaxation sweeps. Improve u^k by one relaxation sweep (see Section 3.2). Concurrently with the sweep, compute some norm e_k of the dynamic residuals (see Section 3.3).

E. Testing the convergence and its rate. If convergence at the current operation level has been obtained $(e_k \le \varepsilon_k)$, go to Step I. If not, and if the relaxation convergence rate is still satisfactory (i.e., if $e_k \le \eta \bar{e}_k$, where η is a prescribed factor which will be discussed below), set $\bar{e}_k \leftarrow e_k$ and go back to Step D (continue relaxation). If the convergence rate is slow $(e_k > \eta \bar{e}_k)$, go to Step F (thus obtain coarser-space correction to u^k).

F. Transfer to coarser level. Decrease k by 1. Introduce, as the first approximation for the new (the coarser) level k, the function

$$u^k = I_{k+1}^k u^{k+1} \tag{3.26}$$

(see discussion of the FAS solution averaging, Section 3.4). Define the S^k problem by computing τ^k as in (3.24b), where the first term is easily calculated using (3.26). As the tolerance for this new problem, set $\varepsilon_k = \delta e_{k+1}$, where δ is a prescribed factor to be discussed below.

G. Finest level stopping parameter. Concurrently with the computation of τ^k, calculate also its norm $\|\tau^k\|$, using the same norm as used for the dynamic residuals (see Step D and Section 3.3). If $k = \ell-1$ set

$$\varepsilon_\ell = \alpha \|\tau^{\ell-1}\| . \tag{3.27a}$$

Usually we want ε_ℓ, the stopping tolerance on the currently finest level, to be comparable to $\|\tau^\ell\|$, so we choose $\alpha = (h_\ell/h_{\ell-1})^\gamma$ (see (3.25b)); or, even more precisely,

$$\alpha = \frac{(h_\ell/h_k)^\gamma}{1 - (h_\ell/h_k)^p} \tag{3.27b}$$

(cf. (3.25b) and (3.25d)).

H. Coarse level solution. If $k = 1$, solve equation (3.24) by relaxation or directly (see Step A above) and go to Step I. If $k > 1$, go to Step C.

I. Employing a converged solution to correct a finer level. If $k = \ell$, go to Step B. If $k < \ell$, make the correction (cf. (3.16))

$$u_{NEW}^{k+1} = u_{OLD}^{k+1} + I_k^{k+1}(u^k - I_{k+1}^k u_{OLD}^{k+1}) \tag{3.28}$$

and then increase k by 1 and go to Step C.

3.7 Comments

3.7.1 Fixed algorithms. The internal checks in the above algorithm can in many cases be replaced by pre-assigned flows. (See for example the fixed algorithm in Figure 3 in Section 4.2.) In particular, instead of checking for slow convergence ($e_k > \eta\bar{e}_k$, in Step E), one can simply switch to the coarser level $k-1$ after a pre-assigned number r_c of relaxation sweeps on level k . The parameter r_c , like η , depends on the smoothing rate (see below). Hence, if that rate is known, r_c may be fixed in advance. Similarly, instead of checking for convergence ($e_k \leq \varepsilon_k$) , the switch to a finer level $k+1$ may be made after a total of r_f relaxation sweeps has been made on level k since the last "visit" to the finer level. (Thus, if $\nu r_c \leq r_f < (\nu+1) r_c$, then ν switches from k to the coarser level $k-1$ are performed before switching to $k+1$.) In some cases $r_f = r_f^k$ depends on k , and in particular it may have special values for $k = \ell$.

The main advantage of fixed algorithms is in saving the work of computing the dynamic residuals at each relaxation sweep. This is a significant saving when the problem is very simple. For example, in the case of the Dirichlet problem (Poisson equation) with linear elements on a uniform square grid (Sec. 4.2), a relaxation sweep costs 5 operations (only additions) per node. A sweep that also computes (3.3) costs 8 operations (7 additions and 1 multiplication) per mode. In more complicated problems, however, the saving is marginal, since a sweep involves many more operations, and computing the norm still adds only 3 operations per nodal value.

In simple problems, where the fixed algorithms are needed, they are as efficient as the "accomodative" ones. In fact, the latter behave like fixed.

3.7.2. Accumulated work units. For comparison purposes, it is common to record the amount of work performed by a multi-grid algorithm in terms of work-units. The work-unit is the computational work of expressing the stiffness (finite difference) equations on the finest level M. A relaxation sweep on level k usually costs $2^{d(M-k)}$ work units, and so does the calculation of the residuals in the transfer from level k to level $k-1$ (except when injection is used). All other computational work (mostly interpolations) is quite small and not easy to express in work units, and is therefore usually neglected. We thus get a measure of the expended work which is independent of the hardware and software being used. This measure is very convenient in comparing various algorithms and in comparing numerical results with theoretical predictions.

We have examined, both experimentally and theoretically, various types of problems (mostly in terms of finite differences), including systems such as the incompressible Navier-Stokes equations. In all cases, the amount of work required to solve the problem to the level of truncation errors (3.27) was between 5 and 10 work-units. In any case, the number of work-units required depends only on the properties of the local operator, and not on the specific data (forcing terms, boundary shape and boundary conditions).

3.7.3 Storage requirements. At any given time, the algorithm has at most one approximation u^k on each level k. Assuming $h_k/h_M \approx 2^{M-k}$, and denoting by $\bar{n}_k$ the amount of storage needed for level k (which is usually a fixed multiple of n_k, the dimension of S^k), it is clear that $\bar{n}_k \approx 2^{d(k-M)}\bar{n}_M$, so that the total storage is less than

$$(1 + 2^{-d} + 2^{-2d} + \ldots)\bar{n}_M = \frac{1}{1 - 2^{-2d}}\,\bar{n}_M \,.$$

This is only a fraction more than $\bar{n}_M$, the storage required for the "target space" S^M.

Moreover, a different variant of the multi-level process requires even far less storage. To see this, note that the finer level $k+1$ is really needed only to provide the τ_i^k correction to the coarser-level equation (3.24a). This correction depends only on the local behavior of the solution. Hence, we never need the entire finer level, neither in core nor in external storage. A segment of it is sufficient for computing τ_i^k (at all points $\underline{x}_i^k$ which are a few mesh-sizes inside that segment).

3.7.4 Optimizations. The effectiveness of any prescribed multi-grid algorithm depends only on local properties (since long error components are reduced, very inexpensively, by coarse-level processes), and can therefore be calculated, once and for all, for any type of functional E, either by local mode analysis or by numerical experiments. In fact, the numerical experiments confirm the mode analysis predictions. We can thus predict which of several possible alternatives will give better performance. We can therefore optimize our algorithm, including the relaxation method (point-wise or line-wise, marching directions, relaxation factors, etc.), the order of interpolations, mesh-size ratios (h_k/h_{k+1}), and switching parameters (η and δ or r_c and r_f).

The relaxation methods and interpolation orders are discussed in Sections 3.1, 3.2 and 3.4 above. The mesh-size ratio $h_k/h_{k+1} = 2$ is close enough to optimum to warrant its general use (at least in two and three dimensional problems).

3.7.5 The Switching parameters. The overall efficiency is not sensitive to the precise choice of δ and η (Steps E and F above). Quite general, good values are $\delta = 0.2$ and $\eta = \max_x \bar{\mu}(x)$, where $\bar{\mu}$ is the smoothing factor of relaxation, i.e., the largest amplification factor (per relaxation sweep) of high-frequency error modes. The high-frequency modes are those which are not

visible in the coarser space; i.e., their projections on the coarser space alias with lower modes. Their amplification factors are easily computed by a local Fourier analysis of the relaxation process (assuming constant triangulation, constant coefficients and infinite domain). $\bar{\mu}$ is a function of x since the triangulation and/or the coefficients may vary over the domain. In well-designed relaxation schemes $\bar{\mu} \approx 0.5$, if not smaller.

Using the switching parameter $\eta = \max \bar{\mu}$ means that relaxation is discontinued and coarse-space correction is sought as soon as the amplitudes of high-frequency residual components are reduced to the level of the lower-frequency amplitudes. Indeed, at this level the smoothing process becomes less efficient (see the comments at the end of Section 3.1), but the error $u^k - U^k$ is already sufficiently smooth.

If coarse space corrections are not efficient enough, μ may always be increased and δ decreased a little, e.g., μ may be replaced by $\mu^{1/2}$ and δ by $\delta/2$. Theoretically optimal values for η and δ are discussed in Appendix A of [B3].

3.7.6 The FAS solution weighting has been discussed above (see Section 3.4). It is important to emphasize that I_{k+1}^k in (3.28), (3.26) and (3.24b) should all be identical. A common programming error is to have them different, at least at some special points. In such programs, the coarse-space corrections will deteriorate as soon as their magnitude becomes comparable to the difference between the different values of $I_{k+1}^k u^{k+1}$.

3.7.7. A case of avoiding relaxation. If u^ℓ is known to be so smooth that an interpolation of order higher than $m_{(\alpha)}+p$ in Step B gives a much better first approximation, then that interpolation should not be followed by relaxation (see footnotes in Sections 3.1 and 3.2). Go from

Step B to Step F. A better procedure, of course, would be to employ a higher approximation order p. If the smoothness is not known in advance and a higher p poses programming difficulties, a better procedure would be to use τ-extrapolations (see below).

3.7.8 Programming. The stiffness equations have the same form (3.24) on all levels. Hence, with a suitable data structure, only one relaxation routine should be written in which the level number (k) is a parameter. The same is true for all other basic operations, such as assembling, τ^k calculations, and interpolations. An example of such a data structure is exhibited in Appendix B of [B3], in Section 4 of [B4] and in the programs of [M78]. The routine for calculating τ^k can be produced easily as a combination of 3 routines: a residual calculation routine (RESCAL), a fine-to-coarse transfer routine (CTF) and a routine (CORSRES) to compute the coarse-space additional term (the first term on the right-hand side of (3.24b)). Both RESCAL and CORSRES are trivial modifications of the relaxation routine (RELAX). The interpolation routines, including CTF, can be written once and for all: they depend on the data structure and the types of elements, but not on the particular functional E. Thus, the programming for each new problem is reduced to the programming of a relaxation routine. A considerable expertise is needed, however, to construct a fully efficient relaxation routine (see Section 3 in [B2], Section 6 in [B5], Lectures 5, 6, 7, 8 in [B6] and [B7]).

3.7.9 Debugging. By printing out every calculated e_k and $\| \tau_l^k \|$, one gets a nice short summary of each run (cf. Appendix B in [B3]). A typical behavior should be exhibited, a deviation from which easily detects most bugs. Detailed debugging techniques (for finite difference formulations) are listed in Lecture 18 in [B6]. Here we should emphasize only one fundamental principle: never settle for any convergence rate slower than (or any work-count

larger than) the prediction of the interior local mode analysis. Due to the iterative character of the method, programming (as well as conceptual) errors often do not manifest themselves in catastrophic results, but rather in considerably slower convergence rates.

3.8 Nonlinear Problems and Continuation (Embedding).

A basic feature of the above algorithm is that it has basically the same efficiency for nonlinear problems as for linear problems. No linearization is required. A difficulty may arise, however, in problems which have more than one local minimum. In such cases, if $S^{\ell-1}$ is too coarse, the first approximation (3.25) may lead to solutions u^{ℓ} converging to a wrong local minimum in S^{ℓ}.

A common way to obtain a first approximation u^{ℓ} close enough to the desired minimum is by a continuation (or "embedding") process: the problem, including its discretization and its approximate solutions, are driven by some parameter γ from an easily solvable (e.g. linear) problem toward the desired problem, in steps $\delta\gamma$ small enough to ensure that the solution to the γ-problem can serve as a good first approximation in solving the $(\gamma+\delta\gamma)$-problem (cf. Section 8.2 in [B3]).

Sometimes the intermediate problems are themselves of interest, i.e., they all correspond to interesting possible states of the physical system being studied. For example, γ may be the total load in a structural mechanics problem, or the Reynolds number in a flow problem, or a parameter in terms of which the boundary conditions, or the shape of the boundary, are expressed, etc. It may then be required to solve the intermediate-γ problems as accurately as the final-γ problem. When the intermediate problems are not of interest, they can be solved to a lower accuracy, using coarser grids. In any case, the grids for the intermediate problems cannot be too coarse, lest the first approximation obtained from them will not lead to the desired minimum. In other words, the intermediate problems should use a sufficiently fine space S^{ℓ}, either because this is their desired accuracy, or because solution components of wavelengths comparable to h_{ℓ} are needed to separate between "attraction regions" of different minima.

Even though the S^{ℓ} high-frequency components are needed in the continuation process, in each $\delta\gamma$ step they do not usually change much. We can therefore employ the "frozen-τ" technique described below, and perform most of the $\delta\gamma$ steps on very coarse spaces, with only a few "visits" to S^{ℓ}. Obtaining a fast approximation by such a continuation process will normally require less computational work than the work in solving (to a better accuracy) the final problem once its first approximation is given.

3.9 Evolution Problems, Optimal-Design Problems, and Frozen-τ Techniques.

We often need to solve not just one isolated problem but a sequence of similar problems depending on some parameter. For example, we may be studying the effect of changing some physical parameters on the "performance" of a system, where the performance is measured in terms of the solution U of the differential problem. We may want to find for what physical parameters the performance is optimal. Or, we may need to solve a sequence of problems in a continuation process (Section 3.8). Or, as the most familiar case, the parameter may be the time t, and the sequence of solutions describe the evolution in time of a physical system. The problem in each time-step, more generally, will be to solve some "implicit" part of the evolution equations.

Such a sequence of problems can be handled very efficiently by multi-level processes. First, we can use the previous solution (or extrapolation from several previous solutions) to obtain a good first approximation for solving the next problem in the sequence. We will then need only one multi-level improvement cycle (involving only a couple of relaxation sweeps on each level) to get a satisfactory solution for each problem in the sequence.

Moreover, by a full use of the multi-level structure, one can usually do much better. One can exploit the very different rates of change of high-frequency and low-frequency components. In parabolic evolution problems, for example, high-frequency components converge to a steady state much sooner than low-frequency ones do. Hence, after the first few steps, the changes in the solution are smooth and can be accurately calculated in increasingly

coarser approximation spaces. (This, incidentally, allows the use of large time-steps without employing implicit equations at all!) Only once in a long time should a step be made in the finest approximation space to readjust the high-frequency components.

Thus, generally, we may like to efficiently solve the next problem in our sequence by neglecting the changes in the high-frequency components (without neglecting the components themselves). The way to freeze the frequencies with wavelengths smaller than $O(h_k)$ is to freeze τ^k in (3.24a), i.e., to solve the next problem with the local-truncation function τ^k that was previously obtained, thereby limiting the current multi-level calculations to the k coarsest spaces $S^1,\ldots,S^k$. This freezes the high-frequency part of the solution but retains its influence on the coarse-grid equations.

Denote by k_j the level of calculation (k) in the j-th step; that is, τ^{k_j} is frozen when solving the j-th problem in the sequence. The sequence of levels k_j can be selected by monitoring the changes in the local-truncation functions τ^k. One method is essentially as follows. Let $\tau^{k,j}$ be the function τ^k at the conclusion of the current step j. Thus, $\tau^{k,j} = \tau^{k,j-1}$ for, and only for, $k \geq k_j$. At each step j we update the quantities

$$\delta_j^k = \| \tau^{k,j} - \tau^{k,i(k,j)} \|, \quad i(k,j) = \max\{i \mid i<j,\ k_i \geq k+2\}, \tag{3.29}$$

using the same norm as in Section 3.3 above. That is, δ_j^k measures the total change in τ^k (owing to calculations at level k+1) since the last calculation at level $k+2$. Since the last visit to S^{k+2}, τ^{k+1} has

remained unchanged. Had we allowed it to change, its changes would roughly be

$$(h_{k+1}/h_k)^{\gamma}\ \delta_j^k = 2^{-\gamma}\ \delta_j^k \quad , \tag{3.30}$$

where γ is defined in (3.25). When (3.30) exceeds a certain tolerance, a new calculation at level $k+2$ is needed, so $k_{j+1} = k+2$ is chosen.

In the case of evolution problems, the contributions of local errors like (3.29) accumulate over time. Instead of δ_j^k, we should then use

$$\hat{\delta}_j^k = \sum_{\alpha=l(k,j)}^{j} \Delta t_\alpha \left\| \tau^{k,\alpha} - \tau^{k,l(k,j)} \right\| \quad , \tag{3.31}$$

where $\Delta t_\alpha = t_\alpha - t_{\alpha-1}$ is the time interval related to the α-th step. Hence, $2^{-\gamma}\ \hat{\delta}_j^k$ is the estimate of the error contributed by the freezing of τ_j^{k+1}. Updating τ_j^{k+1} costs roughly $2^{(2+k-M)d}$ work units, and therefore, the condition when to activate such an update should be

$$2^{-\gamma+d(M-k-2)}\ \hat{\delta}_j^k > \lambda \quad , \tag{3.32}$$

where λ is the marginal rate of exchange between accuracy and work-units. That is, λ is a preassigned constant which represents the smallest profit rate (the smallest added accuracy per work unit) at which we are still willing to invest additional work (see Section 8.1 in [B3]).

Let $M_j = \max_{\alpha \le j} k_\alpha$ be the finest level used up to time t_j in an evolution problem. When (3.32) is satisfied for $k=M_j-1$, it implies the refinement of our system, i.e., the introduction of a new finest level $M_{j+1} = M_j + 1$. Thus, (3.32) serves actually as a <u>discretization-adaptation test</u>. Taking norms like (3.31) not over the entire domain but in small

subdomains, we can in fact apply this test in subdomains, thus deciding not only when to use a finer level but also where to do it.

A more extensive description of adaptation techniques and their discretization patterns is discussed in later chapters.

3.10 τ-Extrapolation

When $S^{\ell-1}$ and S^{ℓ} are "similar" spaces we can use Taylor expansions and express the true local truncation function $\bar{\tau}^{\ell-1}$ as a known multiple of the computed relative truncation function $\tau^{\ell-1}$. Namely,

$$\bar{\tau}_i^{\ell-1} = C_i^{\ell}\tau_i^{\ell-1}\,(1 + o(1)) \tag{3.33}$$

(see (3.25c) - (3.25d) above). Replacing $\tau_i^{\ell-1}$ by its multiple $C_i^{\ell}\tau_i^{\ell-1}$ is a trivial and practically cost-free modification to the above algorithm, which can be conveniently appended to Step G. Such a replacement is called <u>local-truncation extrapolation</u>, or briefly, <u>τ-extrapolation</u>. It makes $U^{\ell-1}$ closer to U (instead of close to U^{ℓ}), and as a result U^{ℓ} will (when corrected by $u^{\ell-1}$ at the next Step I) also be closer to U. In fact, the new truncation error agrees with the true truncation error up to higher-order terms in h, and therefore the solution with τ-extrapolation is <u>equivalent to a higher-order approximation</u>. This can of course be true only in the sense of approximating the nodal values of $I^{\ell}U$ by those of U; in the norm of S, $\|U^{\ell}-U\|$ is as good as any $\|u^{\ell}-U\|$ can be.

Experiments indeed show that if U is smooth enough, then the τ-extrapolated solution u^{ℓ} is much closer to the true solution U than is the full solution U^{ℓ} of the difference equation, namely,

$$\|u^{\ell} - I^{\ell}U\| \ll \|U^{\ell} - I^{\ell}U\| \tag{3.34}$$

In fact, if $I^{\ell}U$ is smooth, (3.34) holds already after the first multi-level cycle to level ℓ (first Step I). See the heuristic explanation in Section 3.5 above and the numerical example in [B5]. If, on the other hand, $I^{\ell}U$ has no smoothness, then the τ-extrapolation will not considerably improve u^{ℓ}. But, exactly in this case, the one multi-level cycle is enough to reduce $\|u^{\ell} - U^{\ell}\|$ well below $\|I^{\ell}U - U^{\ell}\|$, since, exactly in this case, $\bar{\tau}^{\ell}$ is not considerably

smaller than $\bar{\tau}^{\ell-1}$. (The size of the residuals after Step B is roughly $\bar{\tau}^{\ell-1}$, and the one cycle easily reduces them well below $\bar{\tau}^{\ell}$.) So the point of τ-extrapolation is that, after one cycle (bringing the total computational cost typically to 5 to 7 work units), it produces an approximation u^{ℓ} which is guaranteed to be no worse than U^{ℓ} , the full solution of the difference equations, with the nice added feature that any available smoothness is automatically exploited to improve u^{ℓ} even further.

Observe that τ-extrapolation is made at level $\ell-1$, the finest level at which a non-zero approximation to τ is available. The correction is automatically carried over to coarser levels via (3.24b). More precisely, (3.24) is replaced by

$$E_i^k(U^k) = \tilde{\tau}_i^k \quad , \qquad (1 \le k \le \ell) \tag{3.35a}$$

$$\tilde{\tau}_i^k = E_i^k(I_{k+1}^k u^{k+1}) - \sum_j I_{ij}^{k,k+1} \{ E_j^{k+1}(u^{k+1}) - \tilde{\tau}_j^{k+1} \} , \qquad (1 \le k \le \ell-2) \tag{3.35b}$$

$$\tilde{\tau}_i^{\ell-1} = C_i^{\ell} [E_i^{\ell-1}(I_{\ell}^{\ell-1} u^{\ell}) - \sum_j I_{ij}^{\ell-1,\ell} E_j^{\ell}(u^{\ell})] \tag{3.35c}$$

$$\tilde{\tau}_i^{\ell} = 0 . \tag{3.35d}$$

Richardson extrapolation is the classical form of extrapolating in terms of the mesh-size h. It uses the approximate solutions themselves (u^{ℓ} and $u^{\ell-1}$, for example), to produce the extrapolated approximation $(h_{\ell} u^{\ell-1} - h_{\ell-1} u^{\ell})/(h_{\ell} - h_{\ell-1})$. It should be pointed out that the τ-extrapolation can be used in many cases where the Richardson extrapolation cannot. The τ-extrapolation employs Taylor expansions of the local truncation error $\bar{\tau}^{\ell}$, while the Richardson extrapolation requires such an expansion for the global discretization error $U^{\ell} - U$, which is more difficult to get.

4. CONCRETE EXAMPLES

.1 General Linear Elliptic Minimization Problem

Let Ω be any open set in R^d. If $\nu = (\nu_1,\ldots,\nu_d)$ is a multi-index, D^ν will denote the differential operator $D^\nu = D_1^{\nu_1},\ldots,D_d^{\nu_d}$, where $D_j = \partial/\partial X_j$, and $|\nu|$ will denote its order $|\nu| = \nu_1 +\ldots+\nu_d$. The space of real functions defined on Ω whose square is integrable will be denoted by $L^2(\Omega)$. This is a Hilbert space with the scalar product

$$(u,v) = \int_\Omega u(x)\, v(x)\, dx$$

and the norm $\|u\|_o = (u,u)^{\frac{1}{2}}$. The Sobolev space $H^{\dot m}(\Omega)$ is the space of functions in $L^2(\Omega)$ whose derivatives of order less than or equal to $\dot m$ are also in $L^2(\Omega)$, with the scalar product

$$(u,v)_{\dot m} = \sum_{|\nu|\le \dot m} (D^\nu u,\, D^\nu v) \quad ,$$

and norm $\|u\|_{\dot m} = (u,u)_{\dot m}^{\frac{1}{2}}$. For functions $u,v \in H^{\dot m}(\Omega)$, we define a continuous symmetric bilinear functional $a(u,v)$ which is called the strain energy. The ellipticity condition is that $a(u,v)$ is positive definite; namely, there exists an "ellipticity constant" $\alpha>0$ such that

$$a(u,u) \ge \alpha \|u\|_{\dot m}^2 \quad . \tag{4.1}$$

Our space of admissible functions will be the space $S = H_E^{\dot m}$ of functions $u \in H^{\dot m}(\Omega)$ which satisfy the homogeneous essential boundary condition $Bu=0$, where B is a bounded linear operator. (The inhomogeneous condition $Bu=g$ can also be used, with obvious changes in the formalism below.) The general linear d-dimensional elliptic minimization problem of order* $\dot m$ is to find

---

* See footnote on page 55.

U such that

$$E(U) = \min_{u \in H_E^{\mathring{m}}} E(u) , \tag{4.2}$$

where

$$E(u) = \frac{1}{2} a(u,u) - (f,u) . \tag{4.3}$$

It is easy to show that the problem (4.2) - (4.3) has a unique solution U, which must satisfy

$$a(U,v) = (f,v) \quad \text{for every} \quad v \in S . \tag{4.4}$$

When integrated by parts, this equation usually yields an equation of the form

$$(LU,v) = (f,v) \quad \text{for every} \quad v \in S . \tag{4.5}$$

so that U satisfies the differential equation $LU = f$, at least in a weak sense.

Consider now a sequence of approximation spaces $S^1,\ldots,S^M$ of the form (2.2) and typical mesh-sizes $h_1,\ldots,h_M$. As in S, it is easy to see in S^ℓ that $E(u^\ell)$ has a unique minimum U^ℓ, which satisifies

---

*Order here coincides with its definition in [SF], where it is denoted by m. The order of the associated differential operator L (see (4.5)) is $2\mathring{m}$. In finite difference formulations, including [B2] - [B7], the order $m=2\mathring{m}$ of the differential equation is taken as the order of the problem.

$$a(U^{\ell},v^{\ell}) = (f,v^{\ell}) \qquad \text{for every } v^{\ell} \in S^{\ell} . \tag{4.6}$$

The stiffness equations (2.10) here take the form

$$a(U^{\ell},\varphi_j^{\ell}) = (f,\varphi_j^{\ell}), \qquad (j=1,\ldots,n_{\ell}) \tag{4.7}$$

where $\varphi_1^{\ell},\ldots,\varphi_{n_{\ell}}^{\ell}$ are the basis functions of S^{ℓ}. Equation (4.7) is clearly equivalent to (4.6). This linear case can also be written as the linear system of equations

$$\sum_{\gamma=1}^{n_{\ell}} a_{j\gamma}^{\ell} U_{\gamma}^{\ell} = f_j^{\ell} , \qquad (j=1,\ldots,n_{\ell}) \tag{4.8a}$$

where (cf. (2.11))

$$a_{j\gamma}^{\ell} = E_{j\gamma}^{\ell} = a(\varphi_{\gamma}^{\ell},\varphi_j^{\ell}) \tag{4.8b}$$

$$f_j^{\ell} = E_j^{\ell}(o) = (f,\varphi_j^{\ell}) . \tag{4.8c}$$

Hence, the multi-grid system of equations (3.24) takes the form

$$\sum_{\beta} a_{i\beta}^{k} U_{\beta}^{k} = \bar{f}_i^{k} , \qquad (1\le k\le \ell) \tag{4.9a}$$

$$\bar{f}_i^{k} = \sum_{\beta} a_{i\beta}^{k} (I_{k+1}^{k} u^{k+1})_{\beta} + \sum_{j} I_{ij}^{k,k+1} (\bar{f}_j^{k+1} - \sum_{\gamma} a_{j\gamma}^{k+1} u_{\gamma}^{k+1}), \qquad (1\le k\le \ell-1) \tag{4.9b}$$

$$\bar{f}_i^{k} = f_i^{k} , \qquad (k=\ell) \tag{4.9c}$$

where $\bar{f}_i^{k}$ denotes the corrected right-hand side $f_i^{k} + \tau_i^{k}$, and where the ranges of the subscripts are $1 \le i,\beta \le n_k$, $1 \le j,\gamma \le n_{k+1}$. The summations range of course over small subsets, since most of the coefficients $a_{i\beta}^{k}$ vanish.

The Gauss-Seidel relaxation at level k (see (3.2)) is given here by

$$\bar{u}_j^\ell = u_j^\ell - \left(\sum_\gamma a_{j\gamma}^\ell \, u_\gamma^{\ell,j} - \bar{f}_j^\ell \right) / a_{jj}^\ell \quad . \tag{4.10}$$

If τ-extrapolation (3.35) is to be used, equation (4.10b) for $k=\ell-1$ should be replaced by

$$\bar{f}_i^{\ell-1} = f_i^{\ell-1} + C_i^\ell \left\{ \sum_\beta a_{i\beta}^{\ell-1} (I_\ell^{\ell-1} u^\ell)_\beta - f_i^{\ell-} \right.$$

$$\left. + \sum_j I_{ij}^{\ell-1,\ell} \left(f_j^\ell - \sum_\gamma a_{j\gamma}^\ell \, u_\gamma^\ell \right) \right\} \tag{4.11}$$

The equations above are those of the Full Approximation Scheme (FAS). In the linear case, we can also use the Correction Scheme (CS), in which the coarser-space functions are correction functions. Equation (4.9b) is then replaced by

$$\bar{f}_i^k = \sum_j I_{ij}^{k,k+1} \left(\bar{f}_j^{k+1} - \sum_\gamma a_{j\gamma}^{k+1} \, u_\gamma^{k+1} \right) \quad , \quad (1 \le k \le \ell-1) \tag{4.12}$$

while (4.9a) and (4.9c) remain the same. Instead of the FAS correction (3.28) one should of course use the CS correction

$$u_{NEW}^{k+1} = u_{OLD}^{k+1} + I_k^{k+1} \, u^k \quad . \tag{4.13}$$

4.2 The Dirichlet Problem with Linear Elements

The standard example of a linear elliptic minimization problem is the problem (4.2) - (4.3) with the strain energy given by the Dirichlet integral

$$a(u,u) = \int_\Omega \left(\frac{\partial u}{\partial x_1}\right)^2 + \ldots + \left(\frac{\partial u}{\partial x_d}\right)^2 \, dx_1 \ldots dx_d \,, \tag{4.19}$$

and with the homogeneous Dirichlet boundary condition $u(x)=0$ for $x \in \partial\Omega$. (Technically, this boundary condition is introduced by taking $S = H_E^1$ = closure in H of $C_o^\infty(\Omega)$, where $C_o^\infty(\Omega)$ are the infinitely differentiable functions which vanish outside a compact subset of Ω.) In this case the differential operator L in (4.5) turns out to be the Laplace operator

$$-\Delta = -\frac{\partial^2}{\partial x_1^2} - \ldots - \frac{\partial^2}{\partial x_d^2} \,,$$

so that the minimizing function U satisfies the Poisson equation with Dirichlet boundary conditions:

$$-\Delta U = f \quad \text{in} \quad \Omega \,, \tag{4.20a}$$

$$U = 0 \quad \text{on the boundary} \quad \partial\Omega \,. \tag{4.20b}$$

We describe the finite element approximations to the problem in two dimensions (d=2). Let the approximation spaces $S^1,\ldots,S^M$ be spaces of continuous piecewise linear functions based on triangulations as in Figure 2. The triangles touching the boundary may have special forms. Such special triangles are called irregular triangles. A grid point serving as a vertex of one or more irregular triangles is called an irregular point. All other grid points are called regular. Notice that $S^{\ell-1}$ is usually

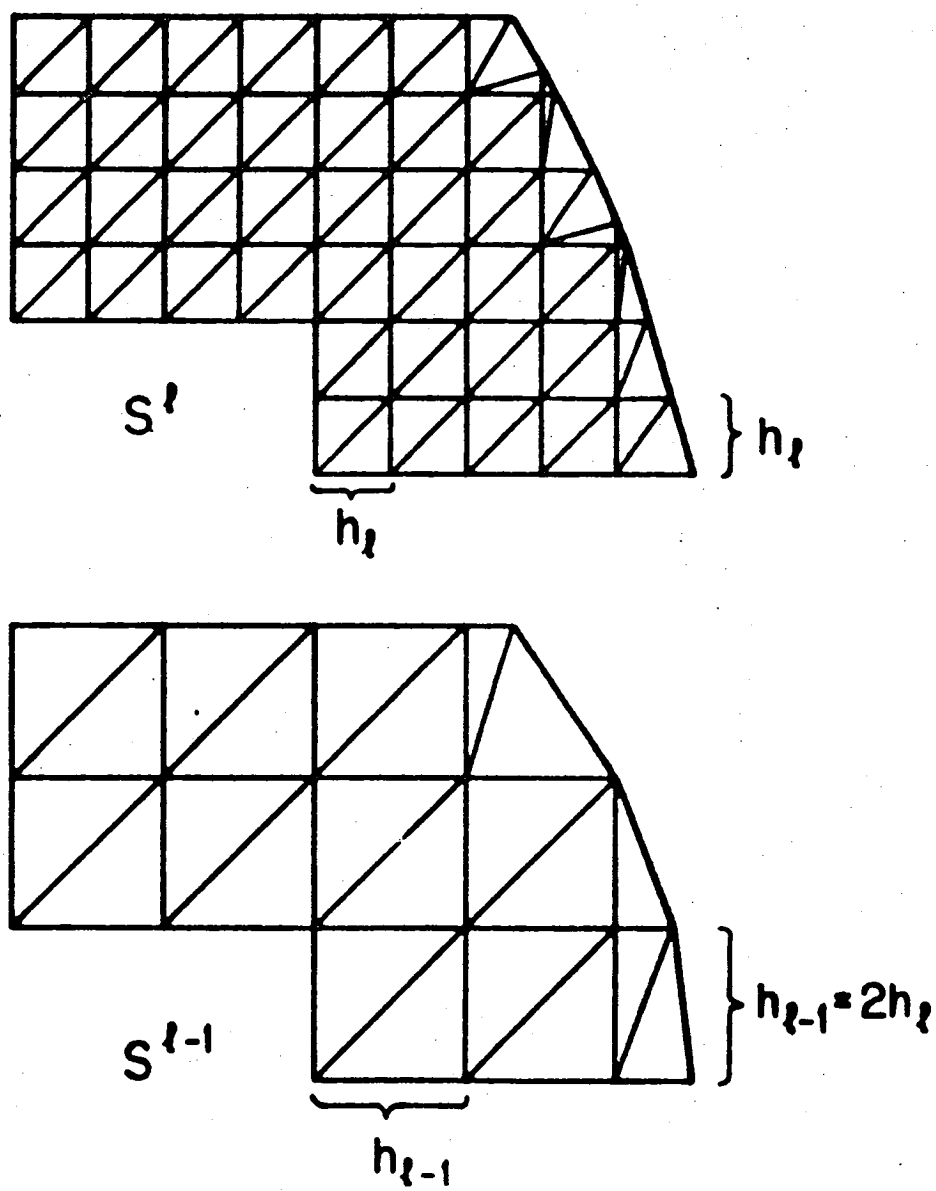


FIGURE 2 Uniform Triangulations

The triangulations are based on uniformly-spaced horizontal and vertical grid lines. The grid lines of S^{ℓ} *are every other grid line of* $S^{\ell-1}$.

not contained in S^ℓ, since its irregular triangles are not unions of S^ℓ triangles. Generally, the nesting condition $S^{\ell-1} \in S^\ell$ would pose a severe limitation on the boundary elements of S^ℓ.

The basis function φ_j^ℓ is the continuous piecewise linear function which vanishes at all grid points, except for $\varphi_j^\ell(x_j^\ell)=1$. It is easy to calculate that, if x_j^ℓ is a regular grid point, then

$$a_{j\gamma}^\ell = a(\varphi_\gamma^\ell,\varphi_j^\ell) = \begin{cases} 4 & \text{if } \gamma=j \\ -1 & \text{if } \gamma \in N_j^\ell \\ 0 & \text{otherwise,} \end{cases} \tag{4.21}$$

where N_j^ℓ is the set of neighbors of j, i.e., $\gamma \in N_j^\ell$ if the distance between x_j^ℓ and x_γ^ℓ is h_ℓ. Hence, at a regular point x_j^ℓ we obtain the difference equation

$$4U_j^\ell - \sum_{\gamma\in N_j^\ell} U_\gamma^\ell = f_j^\ell \ . \tag{4.22}$$

The "volume" of a regular basis function φ_j^ℓ is h^2 (its base area is $3h^2$ and its height is 1), hence $f_j^\ell = (f,\varphi_j^\ell)$ is h^2 times a weighted average of $f(x)$ around x_j^ℓ. Thus (4.22) is h^2 times the usual 5-point approximation to the Poisson equation (4.20a).

Smoothing rate. To calculate the smoothing rate of a Gauss-Seidel relaxation sweep applied to (4.22), we change the notation slightly: If $x_j^\ell = (\alpha h,\beta h)$, where α and β are integers, then we denote U_j^ℓ by $U_{\alpha,\beta}$, and f_j^ℓ by $f_{\alpha,\beta}$, so that (4.22) takes the form

$$4U_{\alpha,\beta} - U_{\alpha-1,\beta} - U_{\alpha,\beta-1} - U_{\alpha,\beta+1} - U_{\alpha+1,\beta} = f_{\alpha,\beta} \ . \tag{4.23}$$

Let $u_{\alpha,\beta}$ denote the value of the approximation u_j^ℓ before the relaxation

sweep, and $\bar{u}_{\alpha,\beta}$ its value after the sweep. If the points $(\alpha h, \beta h)$ are scanned in a lexicographic order, we can write the relaxation equation (3.2) in the form

$$4\bar{u}_{\alpha,\beta} - \bar{u}_{\alpha-1,\beta} - \bar{u}_{\alpha,\beta-1} - u_{\alpha,\beta+1} - u_{\alpha+1,\beta} = f_{\alpha,\beta} . \qquad (4.24)$$

Let us denote the errors before and after the relaxation sweep by $v_{\alpha,\beta} = U_{\alpha,\beta} - u_{\alpha,\beta}$ and $\bar{v}_{\alpha,\beta} = U_{\alpha,\beta} - \bar{u}_{\alpha,\beta}$, respectively. Subtracting (4.24) from (4.23) we get

$$4\bar{v}_{\alpha,\beta} - \bar{v}_{\alpha-1,\beta} - \bar{v}_{\alpha,\beta-1} - v_{\alpha,\beta+1} - v_{\alpha+1,\beta} = 0 . \qquad (4.25)$$

We now apply the local mode analysis. Ignoring boundaries, we can expand the error $v_{\alpha\beta}$ in a Fourier Series

$$v_{\alpha\beta} = \sum_{|\theta| \le \pi} v(\theta)\, e^{i(\theta_1 \alpha + \theta_2 \beta)} , \qquad (4.26)$$

where $\theta = (\theta_1, \theta_2)$ and where we can take $|\theta| = \max [|\theta_1|, |\theta_2|] \le \pi$, since α and β are integers. Similarly,

$$\bar{v}_{\alpha\beta} = \sum_{|\theta| \le \pi} \bar{v}(\theta)\, e^{i(\theta_1 \alpha + \theta_2 \beta)} . \qquad (4.27)$$

Substituting (4.26) and (4.27) into (4.25) we get the amplification factor for the θ component

$$\mu(\theta) = \left| \frac{\bar{v}(\theta)}{v(\theta)} \right| = \left| \frac{e^{i\theta_1} + e^{i\theta_2}}{4 - e^{-i\theta_1} - e^{-i\theta_2}} \right| . \qquad (4.28)$$

The high-frequency components are those with $|\theta| \ge \pi/2$, hence

$$\bar{\mu}(\theta) = \max_{|\theta| \geq \frac{\pi}{2}} \mu(\theta) = .5 \qquad (4.29)$$

is the smoothing factor of the Gauss-Seidel relaxation. That is, each relaxation sweep reduces all high-frequency error components by at worst the factor .5 .

It is possible to ignore boundaries in this analysis, since we only state results concerning high-frequency components. The amplification factors for low frequencies do depend on the boundaries; (4.28) for low-frequencies holds only in some special cases. We can nevertheless deduce from (4.28) that for the lowest frequencies, where $|\theta|=O(h)$, we get $\mu(\theta)=1-O(h^2)$. It means that we would need $O(h^{-2})$ sweeps if we wanted to use relaxation to reduce smooth errors by some factor (e.g., by .5).

Interpolations. If j is a regular point, the natural interpolation (2.5) given by the linear elements of Figure 2 is

$$I_{ij}^{\ell-1,\ell} = \begin{cases} 1 & \text{if } x_j^\ell = x_i^{\ell-1} \\ \frac{1}{2} & \text{if } x_j^\ell = (x_{i1}^{\ell-1} \pm h_\ell,\ x_{i2}^{\ell-1}) \\ \frac{1}{2} & \text{if } x_j^\ell = (x_{i1}^{\ell-1},\ x_{i2}^{\ell-1} \pm h_\ell) \\ \frac{1}{2} & \text{if } x_j^\ell = (x_{i1}^{\ell-1}+h_\ell,\ x_{i2}^{\ell-1}+h_\ell) \\ \frac{1}{2} & \text{if } x_j^\ell = (x_{i1}^{\ell-1}-h_\ell,\ x_{i2}^{\ell-1}-h_\ell) \\ 0 & \text{otherwise} . \end{cases} \qquad (4.30)$$

This linear interpolation is not good enough for the first interpolation (3.1) (Step B in the algorithm). For example, if the solution U is any cubic polynomial, it is easy to see that the nodal values of all the discrete solutions U^ℓ coincide with U. Hence, the first-approximation error

$u^\ell - U^\ell$ is very large compared with the (vanishing) discretization error $U^\ell - I^\ell U$. Moreover, the error $u^\ell - U^\ell$ is highly oscillatory, and hence more expensive to liquidate by multi-level processes. These difficulties do not arise if we replace (4.30) by cubic interpolation, i.e., an interpolation which reproduces every cubic polynomial. Note also that if we measure the error $U^\ell - U$ in L^2 norm, then it does not vanish even if U is a cubic polynomial; $\| U^\ell - U \|_{L^2} = O(h^2)$ notwithstanding the "superconvergence" of the nodal values. Hence, from the point of view of the L^2 error norm, the natural interpolation (4.30) is acceptable.

At later steps in the algorithm (Step I), where <u>corrections</u> are interpolated, the natural interpolation is good enough. Its adjoint can therefore be used as the <u>residual-weighting</u> in Step F (i.e., as $I_{ij}^{k,k+1}$ in (3.24b)). It is interesting to note, however, that a small gain in efficiency is obtainable by using "injection", i.e., residual weighting as in (3.24b) with

$$I_{ij}^{k,k+1} = \begin{cases} 4 & \text{if } x_j^{k+1} = x_i^k \\ 0 & \text{otherwise} . \end{cases} \tag{4.31}$$

Indeed, injection is less expensive, because it requires in Step F the calculation of only one out of four residuals. At the same time the convergence-rate per relaxation work-unit (where the work of Step F is not taken into account) happens to be slightly better with injection than with the residual weighting (4.30).

<u>Local mode analysis</u>. In a manner similar to (but more involved than) the above analysis of relaxation, one can make a Fourier

analysis of the full multi-level cycle, including a fixed number of relaxation sweeps and one coarse-grid correction cycle (see [D1]). Such an analysis shows that with the residual weighting (4.30), and with three relaxation sweeps (per cycle) on each level, the multi-grid convergence factor (i.e., the factor by which at worst errors are reduced) per relaxation work unit is $\mathring{\mu}$ is .59±.02. With injection the factor is $\mathring{\mu} = .547 \pm .015$, which happens to be slightly better. The asymptotic convergence factors observed in the numerical experiments are precisely in these ranges.

If we also include in the work-units count the work of residual weighting (see Section 3.7.2) we get the multi-grid convergence factor per work unit $\tilde{\mu}$. With injection $\tilde{\mu} = \mathring{\mu}^{12/13} \approx .57$, with the residual weighting (4.30) $\tilde{\mu} = \mathring{\mu}^{3/4} \approx .67$.

Incidentally, a good estimate of the convergence factors can be obtained from the smoothing factor $\bar{\mu}$ derived above, whose value is always easier to calculate. The estimate (explained in Section 6.3 of [B3]) is

$$\mathring{\mu} \approx \bar{\mu}^{1-(h_\ell/h_{\ell-1})^d} , \qquad (4.32)$$

which in the present case yields

$$\mathring{\mu} \approx 0.5^{0.75} = .595 .$$

This estimate of course is independent of the residual weighting (and will be approximately realized only when the residual weighting is good enough).

The above convergence factors imply that in one multi-grid cycle the errors are reduced by a factor smaller than .14 (.10 in case of injection). This means that one cycle is actually sufficient, since all we have to reduce the error $\| u^\ell - U^\ell \|$ by is from the

magnitude of $\| U^{\ell-1} - U \|$ (which is its approximate magnitude after Step B) to the magnitude of $\| U^{\ell} - U \|$, which is a reduction by a factor not smaller than .25 .

The one cycle costs less than

$$(3 + \frac{1}{4})(1 + \frac{1}{4} + \frac{1}{16} + \ldots) = \frac{13}{3}$$

work units (cf. Section 3.7.2) in the case of injection (which itself costs $\frac{1}{4}$ work-units). Denoting by W_M the total number of work units needed to solve the S^M problem, we get $W_M = W_{M-1} + \frac{13}{3}$. Since $W_{M-1} = \frac{1}{4} W_M$, we get $W_M = 52/9$.

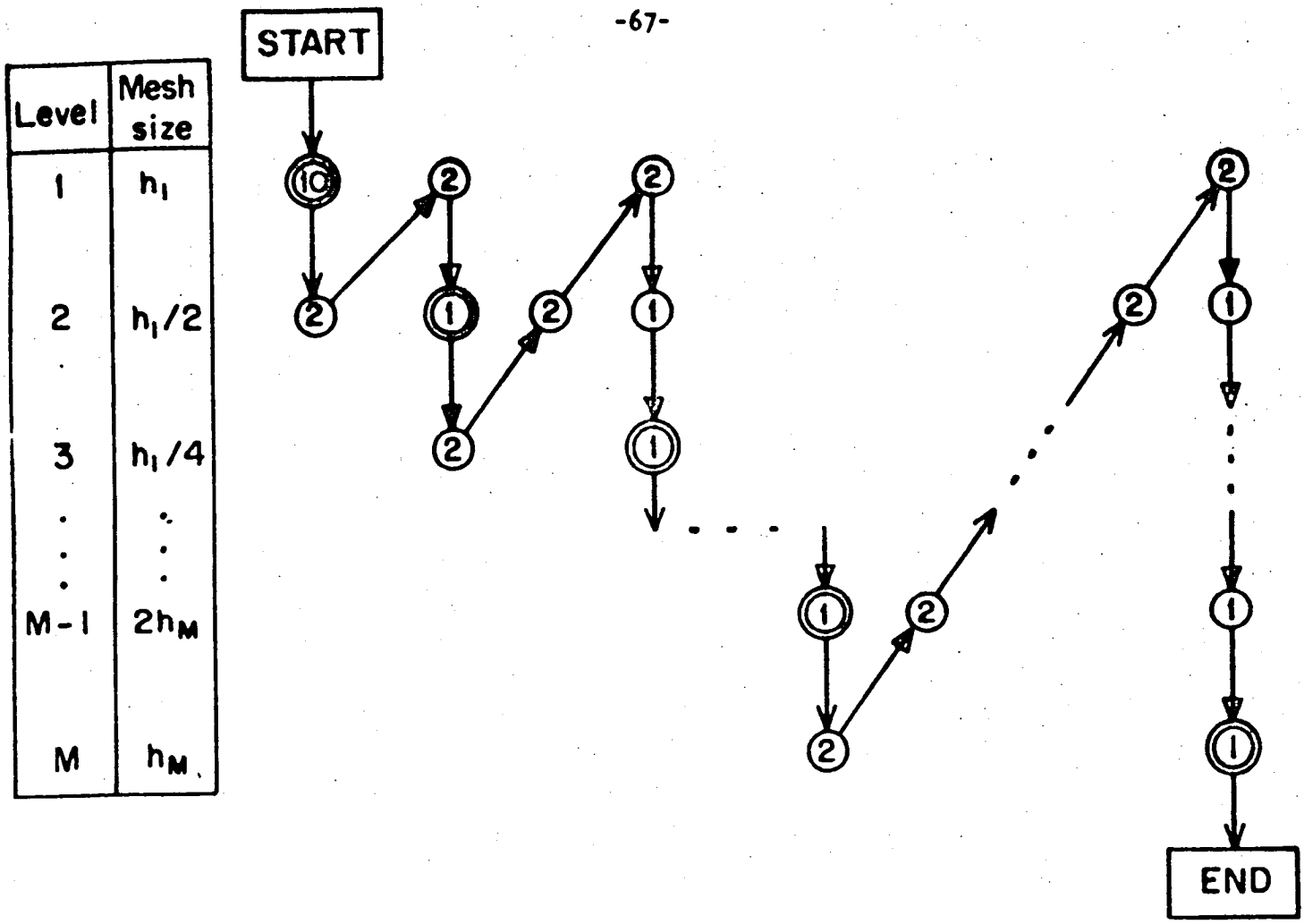
Thus, the multi-level algorithm solves this problem in 5.8 work units.

This estimate is confirmed in many numerical experiments. See for example [B5], where results of further improvements by τ-extrapolation are also exhibited. With careful programming, using the correction scheme and residual injection, this algorithm requires less than $41\, n_M$ additions and no multiplications, where this count includes all operations (unlike the work unit count, which excluded interpolations) and where n_M is the number of degrees of freedom in the finest space S^M (see Section 6.3 in [B3]). The flow of the algorithm is shown in Figure 3. The time measurements of this algorithm which were made by Craig Poling at ICASE in 1977 are summarized in Table 1.

Computer \ Grid	17 x 17	33 x 33	65 x 65	129 x 129	Algorithm
6600	.028 .011	.083 .048	.303 .164	- -	FMG 1 Cycle
Cyber 175	.0117 .0046	.0326 .0161	.1085 .0628	- -	FMG 1 Cycle
STAR 100	-	.0046	.0109	.0347	1 Cycle

TABLE 1 Time Measurements of the Multi-Level Algorithms.

FMG is the Full Multi-Grid algorithm mentioned in the text, and diagrammed in Figure 3. The 1 Cycle algorithm is the same algorithm, but starting when the first approximation is already given in S^M *. The cycle includes a total of 3 Gauss-Seidel relaxation sweeps on each level, except on the STAR 100 computer, where 2 sweeps of Weighted Simultaneous Displacement (see [B3]) were used instead. Residual weighting was made by injection.*



= Cubic interpolation

= Linear interpolation of corrections

= Residual transfer

= Residual transfer where τ-extrapolation can be made

(r) = r relaxation sweeps

◎ = The stage in the algorithm where the error is smaller than the discretization error of that level

Figure 3. Full Multi-Grid Algorithm.

REFERENCES

[B1] Brandt, A. Multi-Level Adaptive Technique (MLAT) for Fast Numerical Solution to Boundary Value Problems. Proceedings of the 3rd International Conference on Numerical Methods in Fluid Mechanics (Paris, 1972), Lecture Notes in Physics 18, pp. 82-29, Springer-Verlag, Berlin and New York, 1973.

[B2] Brandt, A. Multi-Level Adaptive Techniques, IBM Research Report RC6026, IBM T.J. Watson Research Center, Yorktown Heights, N.Y., 1976.

[B3] Brandt, A. Multi-Level Adaptive Solutions to Boundary-Value Problems, Mathematics of Computation, 31, pp. 333-390, 1977.

[B4] Brandt, A. Multi-Level Adaptive Solutions to Partial-Differential Equations — Ideas and Software, Proceedings of Symposium on Mathematical Software (Mathematics Research Center, University of Wisconsin, March 1977), (John Rice, ed.), pp. 277-318. Academic Press, New York, 1977.

[B5] Brandt, A. Multi-Level Adaptive Techniques (MLAT) and Singular-Perturbation Problems, Proceedings of Conference on Numerical Solution of Singular Perturbation Problems (University of Nijmegen, The Netherlands, May — June 1978). Also appears as ICASE Report 78-18.

[B6] Brandt, A. Lecture Notes of the ICASE Workshop on Multi-Grid Methods. With Contributions also by J.C. South (Lecture 8), J. Oliger (10), F. Gustavson (13), C.E. Grosch (14), D.J. Jones (15), and T.C. Poling (16). ICASE, NASA Langley Research Center, Hampton, Virginia, 1978.

[B7] Brandt, A. Multi-Grid Solutions to Flow Problems, Numerical Methods for Partial Differential Equations, Proceedings of Advanced Seminar, Mathematics Research Center, University of Wisconsin, Madison, October 1978. (S. Parter, ed.). To appear.

[BD] Bank, R.A. and Dupont, T. An Optimal Order Process for Solving Elliptic Finite Element Equations, Department of Mathematics, University of Chicago, 1978.

[D1] Dinar, N. On Several Aspects and Applications of the Multi-Grid Method for Solving Partial Differential Equations, NASA Contractor Report 158947, NASA Langley Research Center, Hampton, Virginia (Contract NAS1-14927-3), 1978. This is a preliminary summary of a Ph.D. thesis at the Weizmann Institute of Science, Rehovot, Israel, which will appear in 1979.

[F1] Frederickson, P.O. Fast Approximate Inversion of Large Sparse Linear Systems, Mathematics Report 7-75, Lakehead University, 1975.

[H1] Hackbusch, W. On the Convergence of a Multi-Grid Iteration Applied to Finite Element Equations, Numer. Math. (to appear).

[H2] Hackbusch, W. On the Computation of Eigenvalues and Eigenfunctions of Elliptic Operator by Means of a Multi-Grid Method, Numer. Math. (to appear.)

[H3] Hackbusch, W. An Error Analysis of the Nonlinear Multi-Grid Method of Second Kind, to be published in Aplikace Matematiky.

[M1] Mansfield, L. On the Multi-Grid Solution of Finite Element Equations with Isoparametric Elements.

[M78] MUGTAPE. A Tape of Multi-Grid Software and Programs. Distributed at the ICASE Workshop on Multi-Grid Methods. Contributions by A. Brandt, N. Dinar, F. Gustavson, and D. Ophir, 1978.

[N1] Nicolaides, R.A. On the ℓ^2 Convergence of an Algorithm for Solving Finite Element Equations, Mathematics of Computation, 31, pp. 892-906, 1977.

[N2] Nicolaides, R.A. On Multi-Grid Convergence in the Indefinite Case. ICASE Report 77-12. (To appear in Mathematics of Computation.)

[N3] Nicolaides, R.A. On Some Theoretical and Practical Aspects of Multi-Grid Methods. ICASE Report 77-19. (To appear in Mathematics of Computation.)

[N4] Nicolaides, R.A. On Finite-Element Multi-Grid Algorithms and Their Use. ICASE Report 78-8. ICASE, NASA Langley Research Center, Hampton, Virginia, 1978.

[P1] Poling, T.C. Numerical Experiments with Multi-Grid Methods, M.A. Thesis, Department of Mathematics, The College of William and Mary, Williamsburg, Virginia, 1978.

[S1] Southwell, R.V. Stress Calculation in Frameworks by the Method of Systematic Relaxation of Constraints. I, II. Proceedings of the Royal Society London, Series A 151, pp. 56-95, 1935.

[SB] South, J.C., Jr. and Brandt, A. Application of a Multi-Level Grid Method to Transonic Flow Calculations, ICASE Report 76-8, NASA Langley Research Center, Hampton, Virginia, 1976.

[SF] Strang, G. and Fix, G.J. An Analysis of the Finite Element Method, Prentice Hall, Englewood Cliffs, N.J., 1973.

[Y1] Young, D.M. Iterative Solution of Large Linear Systems, Academic Press, New York, 1972.